SAGE: Style-Adaptive Generalization for Privacy-Constrained Semantic Segmentation Across Domains

Supplementary Material

SAGE enables domain generalization for target models without requiring access to the internal parameters, architecture of the pre-trained models, or large-scale training datasets. This allows the target models to be widely applied across various scenarios. Our research holds significant promise for real-world applications such as autonomous driving, medical image analysis, and remote sensing, where sensitive data and proprietary models are often involved. By achieving strong domain generalization without revealing model internals or requiring private data, our method addresses the increasing demand for privacy-preserving deep learning. SAGE can facilitate the deployment of robust vision systems in areas like healthcare and urban planning, supporting more informed decision-making while safeguarding user privacy. Furthermore, industries relying on third-party models accessed via APIs can benefit from improved performance on unseen domains without compromising security.

The supplementary is organized as follows:

- In Sec. 7, we provide a comprehensive description of the style transfer technique, including implementation details and representative examples of translation.
- Sec. 8 presents the theoretical foundations of the proposed SAGE framework, covering style-prompt generation, adaptive prompt fusion, and the inference process.
- Sec. 9 describes the experimental setup in detail, including dataset specifications, model configurations, training hyperparameters, and computational resources.
- Sec. 10 reports additional experimental investigations, including efficiency comparisons, ablation studies, and extended qualitative segmentation results.
- Sec. 11 discusses the limitations of our approach and outlines potential directions for future research.

7. Style transfer technique

7.1. Implementation details

We adopt a CycleGAN-based [62] approach to perform style augmentation on the source domain. CycleGAN offers powerful unpaired image-to-image translation capabilities, making it well-suited for our domain generalization task. The model consists of two generators, $G_A$and G_B, which learn mappings from the source domain to the target style and vice versa. Each generator is built on a ResNet architecture with 9 residual blocks, allowing effective style transformation while preserving image content.

To improve the realism of generated images, we employ a PatchGAN discriminator that distinguishes between real and translated images. The model is optimized using the original CycleGAN objective, which combines multiple loss terms including adversarial loss, cycle-consistency loss, and identity loss. Training proceeds in two alternating steps: first, updating the generators to fool the discriminators; then, updating the discriminators to better distinguish real from generated images.

We use the Adam optimizer with an initial learning rate of 0.0002 and apply a linear decay schedule. To enhance training stability, we incorporate a history buffer that stores previously generated images for training the discriminator. Model checkpoints are saved every 50 epochs, and intermediate results are visualized to monitor training progress. After training, the style transformation for any input image is performed via a simple forward pass through the generator.

This CycleGAN-based method enables effective style translation on the source domain and provides a strong data augmentation strategy to support our domain generalization framework.

7.2. Examples of translation

Examples of our style augmentation results are illustrated in Figure 7. We select four representative style categories from ImageNet including lakeside, pier, valley, and volcano to simulate distinct environmental appearances commonly seen in real-world driving scenarios. For example: lakeside is used to simulate rainy scenes due to its frequent inclusion of overcast skies, reflections, and blurred contours, mimicking the reduced visibility and color saturation typical of rainy conditions. Pier, often characterized by structured man-made elements like docks and buildings, is used to replicate architectural styles and urban textures. Valley, dominated by open natural landscapes and vegetation, captures the essence of rural or countryside environments. Volcano, with its hazy, low-contrast tones and dramatic lighting, effectively simulates foggy or low-visibility scenes. By performing style transfer using these categories, we can generate stylized versions of the source domain data that mimic various environmental conditions.

8. Theoretical details of SAGE

8.1. Style-prompt generation

Figure 2 provides an overview of the SPG phase architecture. Considering that the target domain may include di-

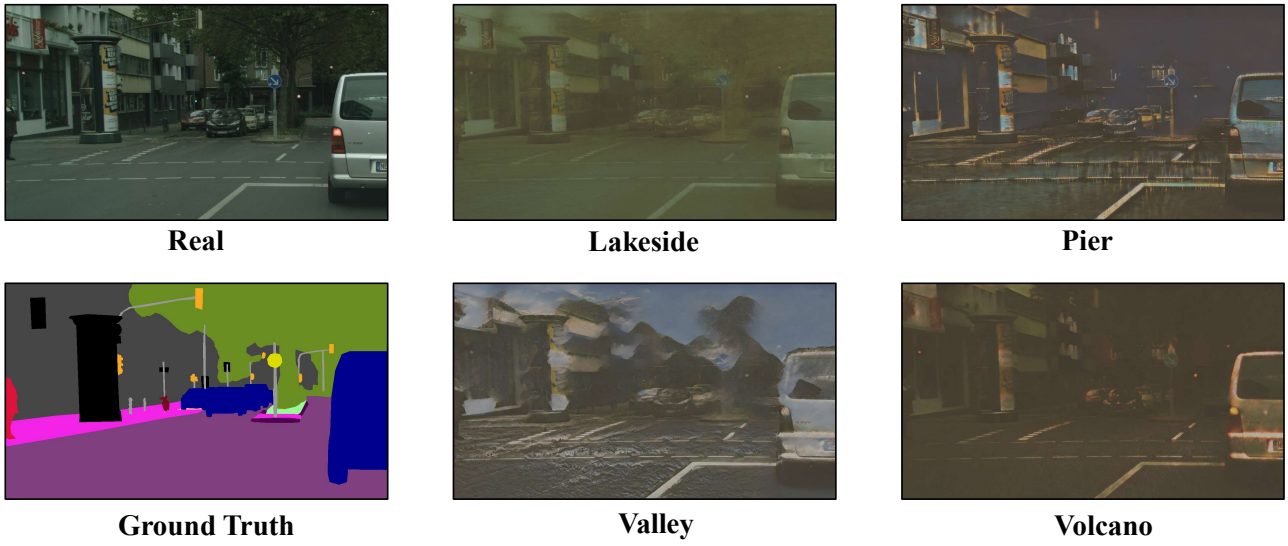

Figure 7. Examples of CycleGAN-based style augmentation.

verse real-world scenes whose visual styles deviate substantially from the source domain, it becomes crucial to generate style-aware prompts that can generalize effectively across such variations. To tackle this challenge, we randomly select n categories from ImageNet [15], which serves as an auxiliary style dataset. These selected categories provide diverse style references. We then employ an CycleGAN translation approach [62] to render source images in the visual style of each reference category.

Nevertheless, visual similarity alone does not guarantee semantic consistency across scenes, prompt relevance may still vary due to object composition and contextual layout. Thus, relying on a single fixed prompt per style reference would be insufficient. To address this limitation, we introduce a dynamic style-prompt generator, which produces content-adaptive prompts tailored to each input image.

Each generator G_i is associated with a particular style $i \in \{1, 2, \ldots, n\}$ and contains a learnable prompt template $\mathcal{T}_i = \{P_t, P_b, P_l, P_r\}$ with learnable weights only located at the four borders (top, bottom, left, and right), while the central region is explicitly zeroed out. This design captures style-related prior knowledge in a spatially constrained form, focusing the modulation on semantically informative boundaries.

To adapt the template to image content, a lightweight modulation network $\mathcal{M}$ is used to compute attention weights that adjust each border prompt. This modulation network is composed of stacked residual blocks, termed ModulatorBlocks, followed by global feature aggregation. The forward function of each ModulatorBlock can be written as:

$$f_{\text{block}}(\mathbf{X}; \theta_i) = \mathcal{F}(\mathbf{X}; \theta_i) + \mathcal{S}(\mathbf{X}; \phi_i) \tag{9}$$

where $\mathcal{F}(\cdot)$ denotes the main convolutional path, while $\mathcal{S}(\cdot)$ is the shortcut pathway, and θ_i, ϕ_i are their respective learnable parameters. Specifically, each main path takes the form:

$$\mathcal{F}(\mathbf{X}; \theta_i) = \text{BN}_2(\text{Conv}_2(\sigma(\text{BN}_1(\text{Conv}_1(\mathbf{X}))))) \tag{10}$$

where σ denotes the ReLU activation function. The network processes the input through three stages with increasing channel dimensions and progressively lower spatial resolutions:

$$\mathbf{X}_1 = f_{\text{block1}}(\mathbf{X}; \theta_1) \in \mathbb{R}^{B \times D_1 \times \frac{H}{2} \times \frac{W}{2}} \tag{11}$$

$$\mathbf{X}_2 = f_{\text{block2}}(\mathbf{X}_1; \theta_2) \in \mathbb{R}^{B \times D_2 \times \frac{H}{4} \times \frac{W}{4}} \tag{12}$$

$$\mathbf{X}_3 = f_{\text{block3}}(\mathbf{X}_2; \theta_3) \in \mathbb{R}^{B \times D_3 \times \frac{H}{8} \times \frac{W}{8}} \tag{13}$$

where $D_1 = d$, $D_2 = 2d$, and $D_3 = 4d$. These representations encode increasingly abstract semantic information. The final stage involves projecting $\mathbf{X}_3$ to a $4C$-channel feature map, then applying adaptive average pooling to yield a compact global descriptor:

$$\mathbf{X}_4 = \text{Conv}_{1\times1}(\mathbf{X}_3) \in \mathbb{R}^{B \times 4C \times \frac{H}{8} \times \frac{W}{8}} \tag{14}$$

$$\mathcal{M}(\mathbf{X}) = \text{AdaptiveAvgPool}(\mathbf{X}_4) \in \mathbb{R}^{B \times 4C \times 1 \times 1} \tag{15}$$

Next, the output is reshaped into modulation coefficients for each border region:

$$\alpha = \text{reshape}(\mathcal{M}(\mathbf{X})) \in \mathbb{R}^{B \times 4 \times C \times 1 \times 1} \tag{16}$$

where $\alpha = [\alpha_p, \alpha_b, \alpha_l, \alpha_r]$ contains the modulation coefficients for each border. The original semantic border $P_t, P_b, P_l, P_r \in \mathbb{R}^{B\times C\times H_p\times W_p}$ are modulated as:

$$\begin{aligned} P_t' &= P_t \odot \alpha_{:,0,:,:,:}, \\ P_b' &= P_b \odot \alpha_{:,1,:,:,:}, \\ P_l' &= P_l \odot \alpha_{:,2,:,:,:}, \\ P_r' &= P_r \odot \alpha_{:,3,:,:,:} \end{aligned} \tag{17}$$

where $\odot$ denotes element-wise multiplication with appropriate broadcasting. Let $Z \in \mathbb{R}^{B\times C\times H_c\times W_c}$ be a zero tensor representing the center region. The final adaptive border prompt is assembled via spatial concatenation:

$$P_i = \begin{bmatrix} P_t' \\ P_l', Z, P_r' \\ P_b' \end{bmatrix} \tag{18}$$

This formulation enables the network to generate style-prompts by adaptively modulating each border region based on the semantic content of the input image.

We then augment the input image $\mathbf{X}$ with style-prompts P_i before feeding it to a pre-trained black-box segmentation model Φ. The augmented input can be formulated as:

$$\mathbf{X}' = \mathbf{X} \oplus G_i(\mathbf{X}) \tag{19}$$

where $\oplus$ denotes the prompt attachment operation and G_i is the style-prompt generator corresponding to style i. The black-box model then produces a predicted segmentation mask:

$$\hat{M} = \Phi(\mathbf{X}') \tag{20}$$

We employ cross-entropy loss $\mathcal{L}_{\text{CE}}$ between the predicted mask $\hat{M}$ and the ground truth mask M to optimize the parameters θ_s of each style-prompt generator:

$$\mathcal{L}_{\text{CE}}(\hat{M}, M) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} M_{i,c}\log(\hat{M}_{i,c}) \tag{21}$$

where N denotes the total number of pixels, and C is the number of semantic categories. After training, each generator G_i is capable of producing effective prompts customized to the specific style it represents.

8.2. Adaptive Prompt Fusion

In the second stage of our framework, we aim to dynamically determine which style prompts are most appropriate for a given target domain image. Despite the availability of diverse style-specific prompts from SPG, real-world images often exhibit a mixture of multiple styles. Therefore, a mechanism that adaptively weighs and combines these prompts is essential.

We introduce the Adaptive Prompt Fusion (APF), which utilizes a cross-attention strategy to compute the relevance between the input image and each candidate prompt. Based on these computed weights, it performs a prompt fusion to enhance segmentation performance across various target styles. Given an input image $\mathbf{X} \in \mathbb{R}^{B\times C\times H\times W}$, we generate a collection of style prompts using n pre-trained generators $G_1, \ldots, G_n$:

$$\mathbf{P}_i = G_i(\mathbf{X}), \quad i \in \{1, 2, \ldots, n\} \tag{22}$$

These are then stacked along a new dimension to form a 5D tensor:

$$\mathbf{P} = [\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_K] \in \mathbb{R}^{B\times N\times C\times H\times W} \tag{23}$$

We then apply L2 normalization over spatial dimensions to reduce scale discrepancies among prompts:

$$\mathbf{P}_{\text{norm}} = \frac{\mathbf{P}}{||\mathbf{P}||_2} \tag{24}$$

To evaluate how relevant each prompt is to the input image, we employ a pre-trained shared encoder $\mathcal{E}$ to extract feature representations. The input image is encoded as:

$$\mathbf{f}_X = \mathcal{E}(\mathbf{X}) \in \mathbb{R}^{B\times D} \tag{25}$$

which is projected into a query embedding space using a learnable projection head:

$$\mathbf{e}_X = W_x(\mathbf{f}_X) \in \mathbb{R}^{B\times d} \tag{26}$$

For the prompts, we first flatten the K variants per image into a batch format:

$$\mathbf{P}_{\text{flat}} \in \mathbb{R}^{(B\cdot N)\times C\times H\times W} \tag{27}$$

Then encode and project them:

$$\mathbf{f}_P = \mathcal{E}(\mathbf{P}_{\text{flat}}) \in \mathbb{R}^{(B\cdot N)\times D} \tag{28}$$

$$\mathbf{e}_P = W_p(\mathbf{f}_P) \in \mathbb{R}^{(B\cdot N)\times d} \tag{29}$$

We reshape $\mathbf{e}_P$ back to group the n prompts per sample:

$$\mathbf{E}_P = \text{reshape}(\mathbf{e}_P) \in \mathbb{R}^{B\times N\times d} \tag{30}$$

Finally, cross-attention scores are computed by dot product between the input query and prompt keys:

$$\alpha_{\text{attn}} = \mathbf{e}_X \cdot \mathbf{E}_P^T \in \mathbb{R}^{B\times N} \tag{31}$$

To derive the final attention weights for prompt fusion, we first normalize the attention scores using a softmax function:

$$\alpha = \text{softmax}(\alpha_{\text{attn}}, \text{dim} = 1) \tag{32}$$

To avoid overconfidence in individual prompts and ensure a smoother distribution, the resulting weights are further compressed via a hyperbolic tangent:

$$\alpha = \tanh(\alpha) \tag{33}$$

The attention weights are reshaped to $\alpha \in \mathbb{R}^{B\times N\times 1\times 1\times 1}$ for broadcasting. The weighted prompts are computed as:

$$\mathbf{P}_{\text{weighted}} = \alpha \odot \mathbf{P} \in \mathbb{R}^{B\times N\times C\times H\times W} \tag{34}$$

where $\odot$ represents element-wise multiplication with broadcasting.

The fused prompt is obtained by summing across the style dimension:

$$\mathbf{P}_{\text{fused}} = \sum_{i=1}^{N} \mathbf{P}_{\text{weighted}}[:, i, :, :, :] \in \mathbb{R}^{B\times C\times H\times W} \tag{35}$$

Finally, the fused prompt is integrated with the input and fed into the black-box model to obtain the predicted results:

$$\mathbf{X}' = \mathbf{X} \oplus \mathbf{P}_{\text{fused}} \tag{36}$$

The separate heads W_x and W_p are optimized based on the loss function in the same manner as Equation 21. This adaptive fusion mechanism allows the model to leverage multiple style prompts simultaneously, weighting them according to their relevance to the input image's characteristics, thereby achieving more effective segmentation across diverse visual domains.

8.3. Inference phase

Algorithm 1: Inference Stage

Input: prompt generators $G_1, G_2, ..., G_n$
shared encoder $\mathcal{E}$
separate heads W_x and W_p
input image x
pretrained model M
Output: Segmentation result y
for $i = 1$ **to** n **do**
  $p_i \leftarrow G_i(x)$
  for *channel* c *in* p_i **do**
    $p_i^c \leftarrow \text{Normalize}(p_i^c)$
$V \leftarrow [p_1; p_2; ...; p_n]$
$[f_{\text{prompt}}, f_x] \leftarrow \mathcal{E}(V, x)$
$K \leftarrow W_p(f_{\text{prompt}})$
$Q \leftarrow W_x(f_x)$
$A \leftarrow QK^\top$
$A \leftarrow \text{softmax}(A)$
$A \leftarrow \tanh(A)$
$V' \leftarrow AV$
$y \leftarrow M(x + V')$
return y

Once the SPG phase and the APF phase are completed, inference on images from an unseen target domain can be performed using the n trained style-prompt generators and the separate heads. The pseudo-code for the inference process is provided in Algorithm 1.

Specifically, given an input image x, we first feed it sequentially into each of the n style-prompt generators G_i to obtain n style-specific prompts. Each channel of these prompts is then normalized using L2 normalization to ensure consistent scale across channels.

Next, the n normalized prompts are concatenated and passed, along with the image x, through the shared encoder $\mathcal{E}$. The resulting features are then split and fed into two separate heads to generate the key (K) and query (Q) matrices.

We compute the attention scores by multiplying Q with the transpose of K, followed by a softmax operation and a subsequent $\tanh$ activation to produce the attention weights for each prompt. These weights are used to compute a weighted sum of the n style-prompts, resulting in the final fused prompt.

Finally, this fused prompt is integrated with the input image and fed into the pre-trained black-box model to obtain the semantic segmentation prediction.

9. Experimental details

9.1. Datasets details

We evaluate our method on five widely-used datasets for domain generalization in semantic segmentation. During training and evaluation, all images from the datasets are randomly cropped and resized to 512×512 resolution.

GTAV(G). GTAV is a synthetic dataset rendered using the Grand Theft Auto V game engine. It contains 24,966 simulated urban scene images with a resolution of 1914×1052, annotated with 19 semantic classes aligned with Cityscapes.

SYNTHIA(S). SYNTHIA is a synthetic dataset containing 9,400 images of resolution 1280×760. Following prior works, we use the SYNTHIA-RAND-CITYSCAPES subset and evaluate on 16 shared categories with Cityscapes. The dataset provides various urban scenes under different environmental conditions.

Cityscapes(C). This real-world dataset is collected from urban street scenes across German cities. It contains 2,975 finely annotated images for training and 500 for validation. The original resolution is 2048×1024, and annotations cover 19 semantic classes.

BDD-100K(B). BDD-100K includes 7,000 training and 1,000 validation images with fine pixel-level annotations across 19 categories. The dataset captures diverse driving scenes under various weather, lighting, and time-of-day conditions. The original resolution is 1280×720.

Mapillary(M). Mapillary provides 25,000 images col-

lected from diverse geographic regions and devices. The images vary in resolution and are captured under a wide range of weather and scene conditions. For consistency with other datasets, we re-map its annotations to the 19 classes used in Cityscapes.

9.2. Model and training hyperparameters

Architecture. In this study, the black-box segmentation model adopts the SegFormer architecture (B5 variant), with pretrained weights sourced from the ADE20K dataset (`nvidia/segformer-b5-finetuned-ade-640-640`). The model structure is implemented using the HuggingFace Transformers library. We adapt the classifier layer in the decode head to align the output number of classes with the Cityscapes dataset. After passing through the backbone encoder, the input image is processed by the decode head to generate logits, which are then upsampled to the original resolution via bilinear interpolation. The modulation network consists of three stacked `ModulatorBlocks` with channel dimensions of 32, 64, and 128, respectively, and a stride of 2 at each layer. A ResNet-18 serves as the shared encoder, initialized with ImageNet-pretrained weights. Its backbone outputs features with a dimensionality of 512, which are then projected to the embedding space (dimension 512) through two fully connected separate heads.

Initialization. The prompt template is initialized using a normal distribution with a mean of 0 and a standard deviation of 0.1 before meta initialization. All convolutional layers in the modulation network are initialized with Kaiming normal initialization (with $a = 0$, `mode='fan_in'`, and `nonlinearity='leaky_relu'`) to match the ReLU activation function. The bias terms of the convolutional layers are initialized to zero.

Optimization and Schedule. During the SPG phase, we use Stochastic Gradient Descent (SGD) as the optimizer, with an initial learning rate of 1×10^{-4} and a momentum of 0.9. A MultiStepLR scheduler is employed to decay the learning rate by a factor of 0.1 at epochs 150, 180, and 210. In the APF phase, we switch to the AdamW optimizer with an initial learning rate of 1×10^{-4} and β parameters set to $(0.5,\ 0.999)$. The learning rate is scheduled using CosineAnnealingWarmRestarts, with $T_0 = 1$, T_{mult} equal to the total number of epochs, and a minimum learning rate of 1×10^{-5}.

9.3. Details of resources used

During the SPG phase, we used approximately 1 hour of GPU time per style-prompt generator on an NVIDIA RTX 4090 GPU, with around 10 GB of memory usage, utilizing the PyTorch library. In the APF phase, training took approximately 4 hours on the same GPU with about 13 GB of memory consumption, also using PyTorch. Most inferences were conducted on an RTX 4090 GPU, with computational costs that were negligible compared to the training phase.

10. More experiments

10.1. Comparison of efficiency

Table 4. Comparison of different methods in terms of parameter count, average training time per iteration, and inference time.

Method	#Params (M)	Training Time	Inference Time
Full Fine-Tuning	84.61	371.32	23.26
SPG Phase	1.00	134.08	-
APF Phase	0.53	272.97	38.09
Total	1.53	407.05	38.09

Table 4 compares our method with Full Fine-Tuning and A2XP in terms of parameter count, average training time per iteration, and average inference time per image. Our model contains less than 2% of the parameters of Full Fine-Tuning, demonstrating its lightweight nature and the fact that it requires no extensive computational resources to update the parameters of the pretrained model. Moreover, our training time is only 35.73 ms longer than that of Full Fine-Tuning, a negligible increase considering the significant reduction in parameters. In addition, our method reduces the training time by 95.53 ms compared to A2XP, another black-box domain generalization method, indicating that the additional steps introduced by our design do not result in a major increase in training time. However, due to the more complex forward pass, our inference time is slightly longer than that of Full Fine-Tuning and A2XP. This is expected and remains within an acceptable range. These results demonstrate that our approach achieves substantial parameter savings while maintaining competitive training and inference efficiency.

10.2. Details of ablation study

Table 5. Comparison of different prompt generator on mIoU performance.

	Trained on G					Trained on S				
	→C	→B	→M	→S	Avg.	→C	→B	→M	→G	Avg.
Border	47.59	38.34	45.98	31.09	40.75	39.96	33.90	35.50	38.49	36.84
A-Border	51.38	39.35	44.12	33.50	**42.09**	41.66	33.91	37.38	38.57	**37.88**
Full	46.60	40.10	43.46	32.89	40.76	42.36	33.52	34.73	36.55	36.79
A-Full	47.26	39.14	43.02	33.77	40.80	40.87	35.91	36.29	37.26	37.58

Table 5 presents the segmentation performance of different generator types under two transfer settings: G → {C, B, M, S} and S → {C, B, M, G}. The style-prompt generator used in Section 3 is denoted as A-Border. A-Full is largely similar to A-Border, except that its prompt template shares the same spatial dimensions as the input image. Its modulation network consists of three cascaded ModulatorBlocks

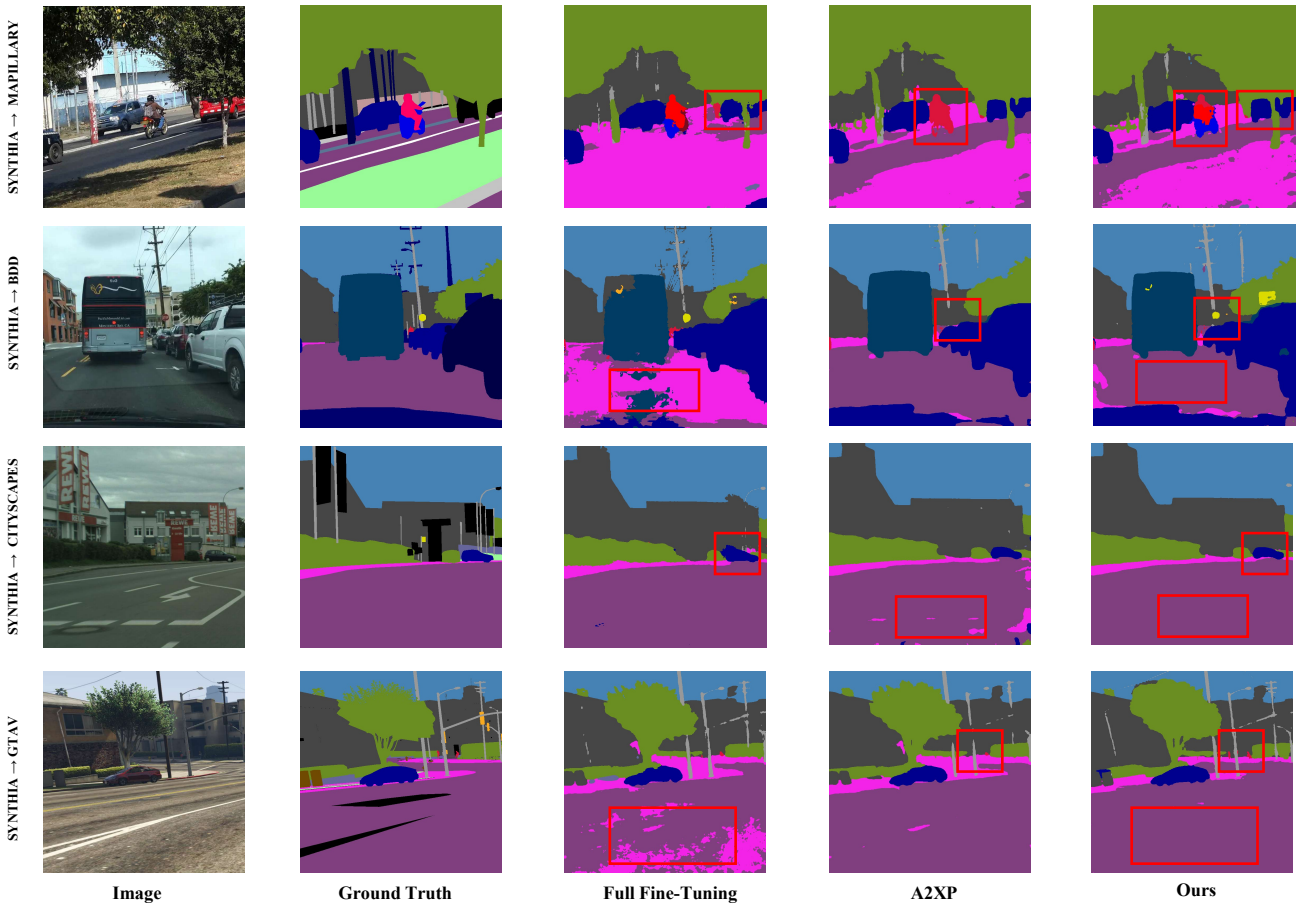

Figure 8. More visual segmentation results under S → {C, B, M, G} setting. The proposed SAGE is compared with Full Fine-Tuning and A2XP. Compared to alternative methods, SAGE yields less noise and demonstrates a better ability to capture details.

followed by a convolutional layer that projects the output to the same number of channels as the input image. This is then upsampled to match the template size, enabling pixel-level reweighting. However, while A-Full introduces additional information, it may also lead to a greater loss of original image features. As a result, it achieves a lower mIoU, which is 1.29% lower than A-Border when trained on domain G. Moreover, for both Border and Full variants, their adaptive versions consistently achieve higher mIoU scores. This validates the effectiveness of our modulation network, which adaptively adjusts different prompt regions according to the input image.

Table 6 shows how different initialization strategies for the prompt template affect segmentation performance. The Meta initialization involves pretraining on subsets of the source domain for each style using a small number of iterations (200 in our setup) to acquire basic prompting capability, followed by style-specific fine-tuning. As shown, Meta initialization consistently achieves the best mIoU across both transfer scenarios. In contrast, the Uniform initialization performs the worst in both settings. This is likely because sampling from a uniform distribution over [0, 1] results in a lack of structured patterns in the initial template, making it more prone to converge in suboptimal directions compared to zero or Gaussian initialization.

Table 6. Comparison of different initialization strategies on mIoU performance.

	Trained on G					**Trained on S**				
	→C	→B	→M	→S	Avg.	→C	→B	→M	→G	Avg.
Zero	46.60	40.10	43.46	32.89	40.76	39.95	36.55	35.22	38.26	37.50
Uniform	47.49	39.66	41.59	32.79	40.38	39.95	35.59	35.39	36.61	36.89
Normal	46.77	41.04	46.07	32.65	41.63	41.80	32.86	36.37	38.32	37.34
Meta	51.38	39.35	44.12	33.50	42.09	41.66	33.91	37.38	38.57	37.88

We evaluate the impact of different ImageNet style references and the number of styles n on generalization performance. As shown in Table 7, SAGE is robust to various style combinations, with $n = 4$ (L, P, V, Vo) achieving the optimal average mIoU of 43.90%. Notably, increasing n beyond 4 does not yield further significant gains, while

Table 7. The sensitivity analysis on style reference. Style abbreviations: L=lakeside, P=pier, V=valley, Vo=volcano, A=ambulance, S=shark, B=barometer, C=can_opener, Sn=snorkel, T=tennis_ball.

Styles	n	**Trained on C**					**Styles**	n	**Trained on C**				
		→B	→M	→G	→S	Avg.			→B	→M	→G	→S	Avg.
L,P,V	3	41.46	39.39	40.86	31.34	38.26	B,P,Sn,T,Vo	5	40.04	41.04	39.78	31.25	38.03
L,P,V,Vo	4	**45.85**	**50.27**	**47.07**	**32.40**	**43.90**	B,C,S,Sn,T	5	40.77	42.85	40.67	31.33	38.91
A,L,S,V	4	36.98	39.60	36.46	29.36	35.60	L,P,Sn,V,S	5	40.87	43.21	39.95	31.42	38.86
P,T,V,Vo	4	41.43	40.63	39.61	31.45	38.28	S,P,B,C,Vo	5	40.65	45.28	40.26	30.61	39.20
A,B,C,Vo	4	41.90	43.90	39.27	31.77	39.21	A,C,L,P,V,Vo	6	43.41	42.07	39.55	31.28	39.08
A,Sn,T,V	4	39.59	43.04	38.80	30.94	38.09	B,C,L,Sn,T,Vo	6	40.83	40.90	40.59	31.47	38.45

inference latency remains independent of n due to our parallelized prompt generation and fusion mechanism.

10.3. Visual segmentation results

Figure 8 presents additional segmentation results under the S $\rightarrow$ {C, B, M, G} transfer setting, comparing Full Fine-Tuning, A2XP, and our method. It is evident that Full Fine-Tuning produces more noisy predictions, leading to issues such as overlapping boundaries and incomplete segmentation. For instance, in domains M and C, the car boundaries are poorly defined, and in domains B and G, the road areas contain significant noise. Although A2XP shows cleaner segmentation with more accurate boundaries, it tends to overlook small objects in the scene. For example, it fails to distinguish between the person and the motorcycle in M, misses the traffic sign in B, and completely ignores a pedestrian in G. In contrast, our method consistently maintains low-noise segmentation while effectively capturing fine details across all four domains, demonstrating its superior ability to balance boundary clarity with object completeness.

11. Limitation and future work

Despite the effectiveness of SAGE in generalizing to target models without accessing the internal parameters, architecture, or large-scale training datasets of pre-trained models, several limitations remain. For instance, although our method retains the black-box setting by relying on the backbone's strong feature extraction capabilities, it still requires access to the final classifier of the target model, with its number of classes adjusted to match the actual dataset. Moreover, the black-box model is required to return not only segmentation outputs but also gradients, which are essential for updating the external style-prompt generator.

To address these limitations, our future work will explore adaptation under fully black-box settings, along with gradient-free optimization strategies for prompt generation. In addition, we plan to extend our method to support domain generalization with multi-modal data and enhance its robustness to out-of-distribution (OOD) samples.