Trans-Adapter: A Plug-and-Play Framework for Transparent Image Inpainting

Supplementary Material

A. Potential Applications

Transparent Image Editing. By not inpainting from the pure noise, our method can also serve as a transparent image editor naturally. As shown in Fig. 1, users can plot color strokes on the original transparent image. These color strokes will be considered as a mask. Then, we add noise to the drawn RGB and alpha map with a strength of 0.99 as the initial noise. Finally, we perform denoising following the previous approach to obtain the edited result.

Extending to Community Models. Since both BrushNet and our Trans-Adapter are plug-and-play modules, they can be applied together to other community models, enabling inpainting based on different models. As shown in the Fig. 2, by using different community models, we can achieve inpainting effects in various styles.

ControlNet Extension. As shown in Fig. 3, we demonstrate that existing control models like ControlNet [5] can be applied to our model for enriched functionality. Since ControlNet does not provide a control model for SD-Inpainting and only supports T2I generation, we apply Trans-Adapter to BrushNet based on SD1.5 and then use ControlNet (with Scribbles) to control image details. The visualization results demonstrate that our method can effectively integrate ControlNet to control inpainting outputs, allowing users to guide structure and details more precisely.

Figure 1. Demonstration of color-stroke-based transparent image editing. Users can draw on a transparent image and obtain an estimated result based on their strokes and a provided text prompt.

Figure 2. While combining with BrushNet, our Trans-Adapter also supports community models for inpainting.

B. More Details of AEQ Assessment

Data Augmentation. We collect a high-quality transparent image dataset with clean alpha edges with 1,000 images from the online PNG stock and matting data. Starting from these transparent images, we simulate images with varying degrees of edge quality to create a comprehensive training set. To generate low-quality edges, we: (1) set regions with an alpha value lower than 50 to a solid color, then perform a dilation operation or a Gaussian blur operation on the alpha map. For the expanded region at the alpha map, we multiply it with the mask generated by the fractal noise to create a more realistic edge degradation effect. (2) composite the images with different backgrounds (solid color backgrounds + natural scenery backgrounds) and then process these images using different segmentation and matting methods [3, 4]. To quantify edge degradation, we compute the difference between the original image's alpha map and its augmented counterpart, identifying regions with significant discrepancies as low-quality edges. In this way, we can train a segmentation network for alpha edge quality assessment.

Training Loss. Since the edge quality assessment is a binary classification task, we use a weighted cross-entropy loss to address class imbalance, as low-quality edge pixels are much fewer than high-quality ones. We assign a higher weight to the low-quality class and further emphasize edge regions using an edge mask $\mathcal{M}_e$.

Here, I_{concat} denotes the input, $\mathcal{F}$ is the segmentation network, y is the ground truth label map, and $\mathcal{M}_e$ is the edge mask. The loss is defined as:

$$\mathcal{L} = \frac{1}{HW} \sum_{i=1}^{HW} \text{CE}(\mathcal{F}(I_{\text{concat}})_i, y_i, w_{y_i}) \cdot (1 + w_e \mathcal{M}_e) \quad (1)$$

where w_{y_i} denotes the class weight (w_0 for high-quality, w_1 for low-quality, typically w_0 as 1.0, w_1 as 10.0), and w_e is the edge weight (set to 4.0). This loss encourages the model to focus more on low-quality edge regions during training.

AEQ Visualization. To demonstrate the effect of our proposed AEQ, we visualize the estimated artifact maps for images in various styles. As shown in Fig. 4, AEQ effectively highlights boundary artifacts across different image types, indicating its strong generalization ability beyond specific styles.

Network Architecture and Training Details. We adopt a lightweight U-Net-based segmentation network comprising three downsampling and three upsampling blocks. Each block contains two convolutional layers with ReLU activation and batch normalization. The network takes an 8-

Table 1. Quantitative comparison of different LoRA training strategies under pure noise and blended noise settings.

	Pure Noise				Blended Noise			
Method	AS↑	LPIPS↓	CLIP Sim↑	AEQ↑	AS↑	LPIPS↓	CLIP Sim↑	AEQ↑
Ours	**6.025**	**0.0591**	26.870	**0.9871**	**6.097**	0.0408	27.030	**0.9878**
Frame-specific LoRA	5.992	0.0616	27.165	0.9855	6.089	0.0416	27.036	0.9863
Frozen LoRA	5.979	0.0637	27.043	0.9817	6.067	0.0471	27.005	0.9831
Frozen Frame-specific LoRA	5.985	0.0624	27.166	0.9856	6.082	**0.0408**	27.023	0.9859
w/o LoRA	5.982	0.0769	**27.363**	0.9863	6.091	0.0430	**27.103**	0.9868

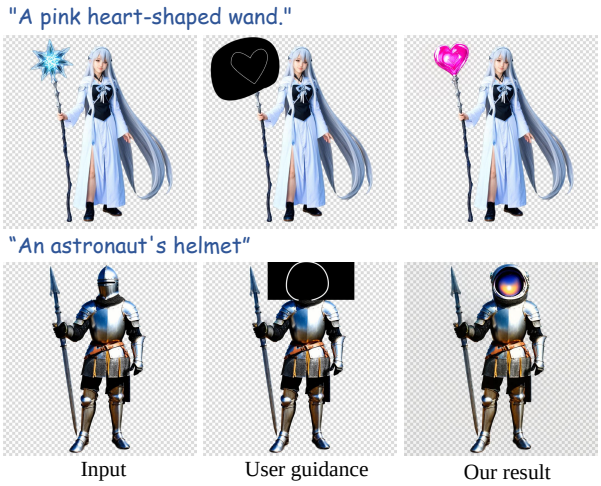

Figure 3. Our approach can be directly combined with control models like ControlNet [5] to enhance functionality. Users can define a mask and outline the inpainting region to generate a transparent image.

channel input and produces a 2-channel output, with hidden channel dimensions of 64, 128, and 256, respectively. We use the Adam optimizer with a learning rate of 1×10^{-4}. Training is performed on images with resolutions of 512×512 and 1024×1024, using a batch size of 4 and randomly selecting a resolution for each batch. The network is trained for 40,000 iterations.

C. More Details of Trans-Adapter

Stage 1: Alpha Map LoRA Training. To enable the generation of large areas of pure black and pure white in the alpha map during inpainting, we adopt offset noise [2] with a weight of 0.1 during LoRA training, as conventional finetuning often struggles with such cases. The LoRA rank is set to 16 and the LoRA alpha is set to 32. We use the AdamW optimizer with an initial learning rate of 1×10^{-4}, and train for 40,000 steps with a batch size of 4.

Stage 2: Joint Finetuning. In the joint finetuning stage, we first load the pretrained LoRA weights, then zero-initialize the spatial alignment module and cross-domain self-attention module. These two modules are finetuned with a learning rate of 5×10^{-5} using the AdamW optimizer for 100,000 steps, with a batch size of 4. We conduct

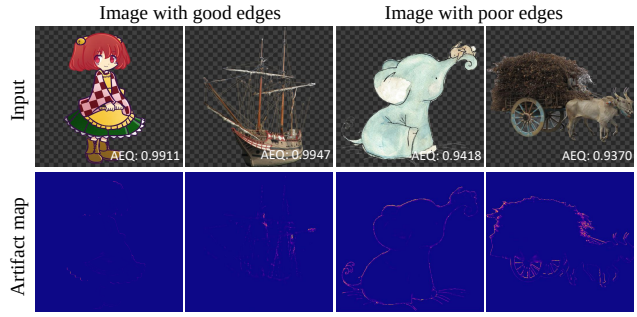

Figure 4. Estimated artifact map of our proposed AEQ for images in different styles. (Zoom in for better visualization.)

an ablation study to compare different training strategies, including frame-specific LoRA (training LoRA only on alpha images, affecting only the alpha channel), frozen LoRA (training LoRA on both RGB and alpha images, then freezing LoRA during stage 2), frozen frame-specific LoRA, and training without LoRA (directly training stage 2 without LoRA). As shown in Table 1, our full method achieves the best overall performance in terms of AS, LPIPS, and AEQ metrics under both pure noise and blended noise settings. Our method achieves the best AEQ and LPIPS, validating the effectiveness of the two-stage training. Removing or freezing LoRA degrades AEQ and LPIPS, highlighting the importance of LoRA-based alpha map pretraining.

D. Limitations

While our proposed method achieves strong performance in transparent image inpainting and editing, several limitations remain. First, our approach is based on SD1.5 and SDXL, which are known to struggle with generating realistic faces and hands during inpainting. This can result in artifacts when editing or restoring these regions, limiting applicability in scenarios requiring high-fidelity human features. Second, when the strength is set to 1.0 for SDXL-inpainting 0.1, the quality of the generated image degrades, which also affects our RGBA inpainting results. This limitation originates from the base checkpoint. Finally, the alpha maps in the MAGICK dataset [1] used for training are not perfect; some regions, such as eyes, are translucent. As a result, our model may also produce undesired translucent areas in these regions during inpainting.

References

[1] Ryan D Burgert, Brian L Price, Jason Kuen, Yijun Li, and Michael S Ryoo. Magick: A large-scale captioned dataset from matting generated images using chroma keying. In *CVPR*, 2024. 2

[2] Nicholas Guttenberg. Diffusion with offset noise. https://www.crosslabs.org/blog/diffusion-with-offset-noise, 2023. 2

[3] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023. 1

[4] Jiachen Li, Jitesh Jain, and Humphrey Shi. Matting anything. In *CVPR*, 2024. 1

[5] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023. 1, 2