Watermark-embedded Adversarial Examples for Copyright Protection against Diffusion Models

Supplementary Material

A. Image Generation under Other Scenarios

A.1. DreamBooth

In this experiment, we evaluate the performance of our method on DreamBooth [43] which is another method to personalize text-to-image models. Given a few (3-5) images of a subject, DreamBooth fine-tunes the pre-trained text-to-image model to learn a unique identifier for that subject. After the fine-tuning, we can use the unique identifier to generate contextualized images of the subject in different scenes, poses, and views. We perform the DreamBooth fine-tuning by using the Python library diffusers [5] on both the original images and adversarial examples. We set the resolution to 512, the learning rate to 2×10^{-6} and the maximum number of train steps to 2000. The steps and parameters used to generate our adversarial example are the same as those used in other image generations in Section 3.4.

Visualization examples are shown in Figure 7. We compare the results generated from the original images with those from our adversarial examples. For the original images, the features of the input images can be learned and used to generate new images. On the other hand, after applying our attack, images are generated with visible watermarks and chaotic textures. Therefore, our method is also applicable to DreamBooth settings to protect copyrights.

A.2. LoRA Fine-tuning

Low-Rank Adaptation (LoRA) [20] is a method initially proposed for fine-tuning large-language models. It freezes pre-trained model weights and injects trainable layers in each transformer block. In the case of fine-tuning diffusion models, LoRA can be applied to the cross-attention layers that relate the image representations with their corresponding prompts. We perform the LoRA fine-tuning by using the Python library diffusers on both the original images and our adversarial examples. We set the resolution to 512, the learning rate to 10^{-4}, the maximum gradient norm to 1.0 and the maximum number of train steps to 2000.

Visualization examples are presented in Figure 8. We compare the results generated from the original images with those generated using our adversarial examples. For the original images, although the effectiveness of LoRA fine-tuning is not as pronounced as that of textual inversion and Dreambooth, the features of the input images can still be extracted to generate new images. On the other hand, after applying our attack, the resulting images exhibit visible watermarks and chaotic textures. Consequently, our method could be effective in LoRA fine-tuning settings for copyright protection purposes.

[5]https://github.com/huggingface/diffusers

A.3. Custom Diffusion

Custom Diffusion [4] is another fine-tuning technique for personalizing image generation models. Given a few (4-5) example images, Custom Diffusion works by only training weights in the cross-attention layers, and it uses a special word to represent the newly learned concept. We perform the Custom Diffusion by using the Python library diffusers on both the original images and our adversarial examples. We use single-concept fine-tuning and set the resolution to 512, the learning rate to 10^{-5} and the maximum number of train steps to 250.

Visualization examples are presented in Figure 9. We compare the results generated from the original images with those generated using our adversarial examples. For the original images, the features can be learned and utilized to generate new images. In the case of the adversarial examples, although no explicit watermark is visible on the generated images, many generated images contain irrelevant text, and many generated content appears distorted and unnatural. Despite the outcome being different from our intended target, our method still demonstrates potential for preventing image imitation under the Custom diffusion.

B. Implementation Details of the Model

Generator architecture We adopt the naming conventions established by [22, 64]. Here, dk refers to a 3×3 Convolution-InstanceNorm-ReLU layer with k filters and stride 2. The notation Rk represents a residual block that contains two 3×3 convolution layers with the same number of k filters. Lastly, uk signifies a 3×3 fractional-strided-Convolution-InstanceNorm-ReLU layer with k filters and a stride of 1/2.

The generator is composed of the following layers:

- The encoder consists of a series of downsampling layers: $d64, d128, d256$.
- This is followed by a series of residual blocks: $R256, R256, R256, R256$.
- The decoder then upsamples the feature maps: $u128, u64$.
- The final layer of the generator is a 3×3 Convolution-Tanh layer with the number of output channels corresponding to the image's number of channels.

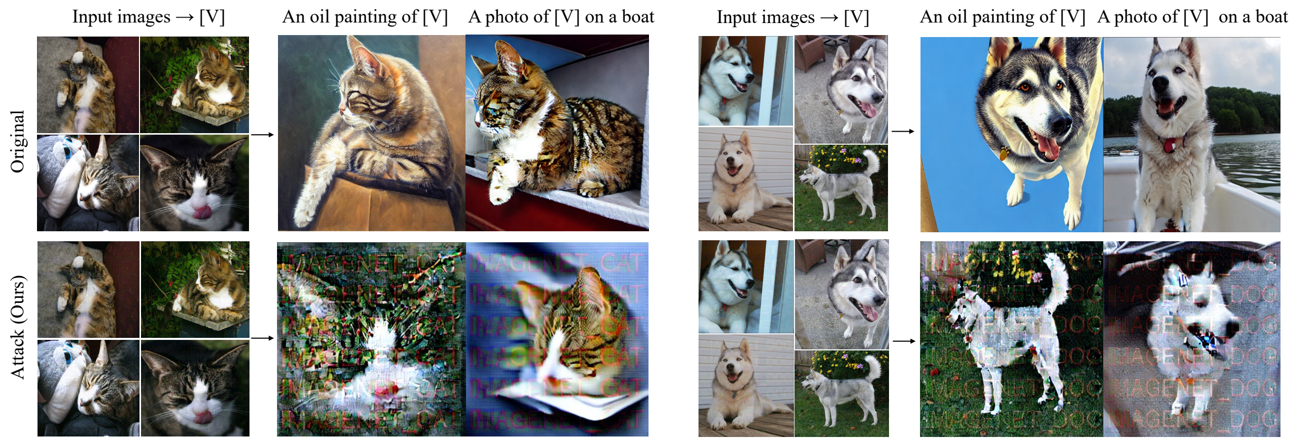

Figure 7. Comparison between the original result and our attack result under DreamBooth on ImageNet. The watermarks used in the examples are IMAGENET_CAT and IMAGENET_DOG.

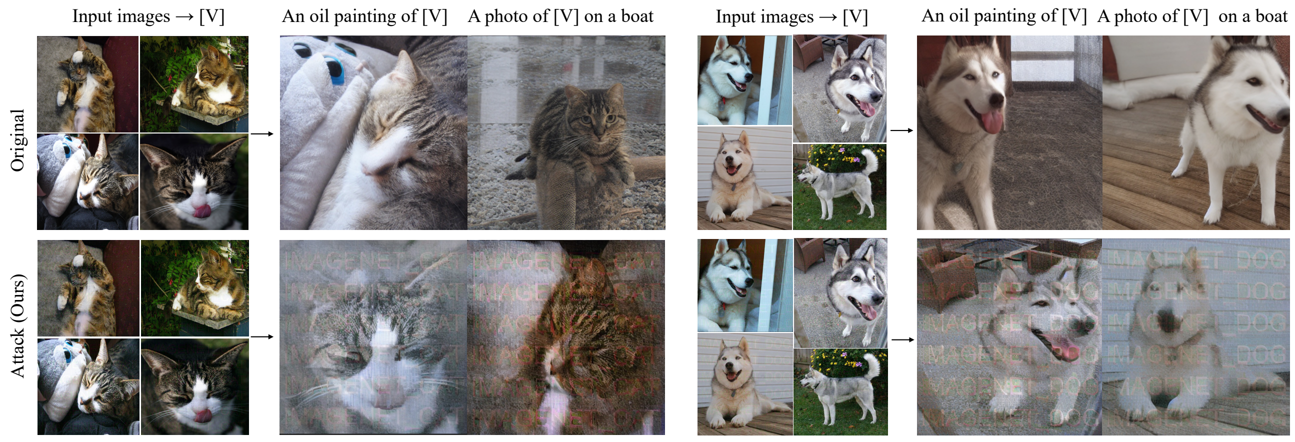

Figure 8. Comparison between the original result and our attack result under LoRA fine-tuning on ImageNet. The watermarks used in the examples are IMAGENET_CAT and IMAGENET_DOG.

Discriminator architecture We utilize a convolutional neural network (CNN) structure inspired by [40]. The notation Ck indicates a 4×4 Convolution-LeakyReLU layer with k filters and stride 2. We do not apply Instance Normalization to the first layer of the discriminator. Leaky ReLUs are used with a negative slope of 0.2.

The discriminator is composed of the following layers:

- A series of convolutional layers with increasing filter sizes: $C64, C128, C256, C512, C1024$.
- The final convolutional layer is a 16×16 Convolution layer that outputs a single feature map.
- A Sigmoid activation function is applied to the output of the last layer to obtain a probability value indicating whether the input image is original or adversarial.

C. More Results of Text-guided Image-to-image Generation

C.1. Results under Different Strength Value

The strength parameter plays a crucial role in image-to-image generation. It determines the level of noise that is added to the original image while generating new images. A small strength value will produce an image nearly identical to the original, while a large strength value will produce an image that largely differs from the original. We explore the change in the watermark on the generated images with different strength values. As shown in Figure 10, NCC remains at a high value when the strength is smaller than 0.4 and drops after that. We also observe that when the strength is increased to 0.55, the content of the generated image is already far from the original one, which makes it difficult

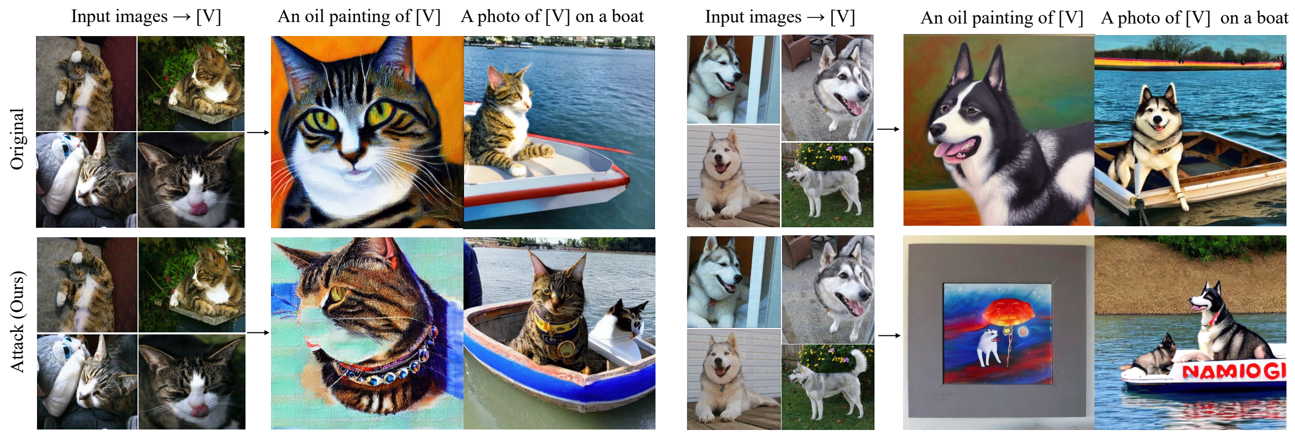

Figure 9. Comparison between the original result and our attack result under Custom Diffusion on ImageNet. The watermarks used in the examples are IMAGENET_CAT and IMAGENET_DOG.

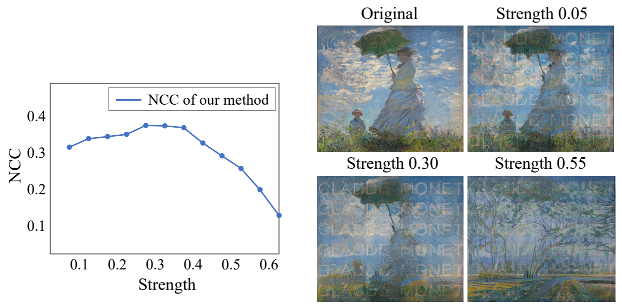

Figure 10. The influence of strength parameter on the visibility of watermark under image-to-image generation.

to identify as a copyright violation. Therefore, we focus on the setting with a strength smaller than 0.5, and our method can successfully attack DMs under this setting.

C.2. Results under Varied Prompts

For text-guided image-to-image generation, the ideal approach is to use varied prompts to guide the generation of new images. However, with thousands of diverse test images, preparing individual prompts for each image is time-consuming. Consequently, we used a uniform prompt for both training and inference.

On the other hand, to explore the influence of different prompts, we evaluated 40 images with 5 varied prompts per image. The Normalized Cross-Correlation (NCC) results between the generated images and the watermark were similar for both fixed prompts (0.31) and varied prompts (0.30). This similarity may be due to the strength parameter being the primary determinant of the level of noise added, while the prompts merely guide the noise during generation. Therefore, even with varied prompts, the watermarks on the generated images remain almost unchanged when the strength parameter is constant.

D. More Visualization Examples

D.1. Text-guided Image-to-Image Generation

We show more examples under image-to-image generation in Figure 11. The experimental settings remain consistent with those in Section 4.2. Compared to the existing methods that only add chaotic content, our method adds visible watermarks, providing a more straightforward way to show copyright violations.

D.2. Textual Inversion

We show the comparison result between our method and the previous methods under textual inversion in Figure 12. The experimental settings remain consistent with those in Section 4.3. The previous methods could encourage DMs to generate images that are far from the original ones (AdvDM) or with large artifacts (Mist). However, such changes are not straightforward enough to indicate copyright violations. Our method succeeds in compelling DMs to generate images with obvious watermarks, demonstrating a simple yet powerful way to prevent copyright violations.

D.3. Transferability on Other Generative Models

We show more visualization examples of this setting in Figure 13. All examples follow a similar trend in that our method can add visible watermarks to the generated images for most models. For Runway, although there is no obvious watermark, our method still forces the models to generate images that differ from the original ones. Our method exhibits good transferability and can attack various image-generation models.

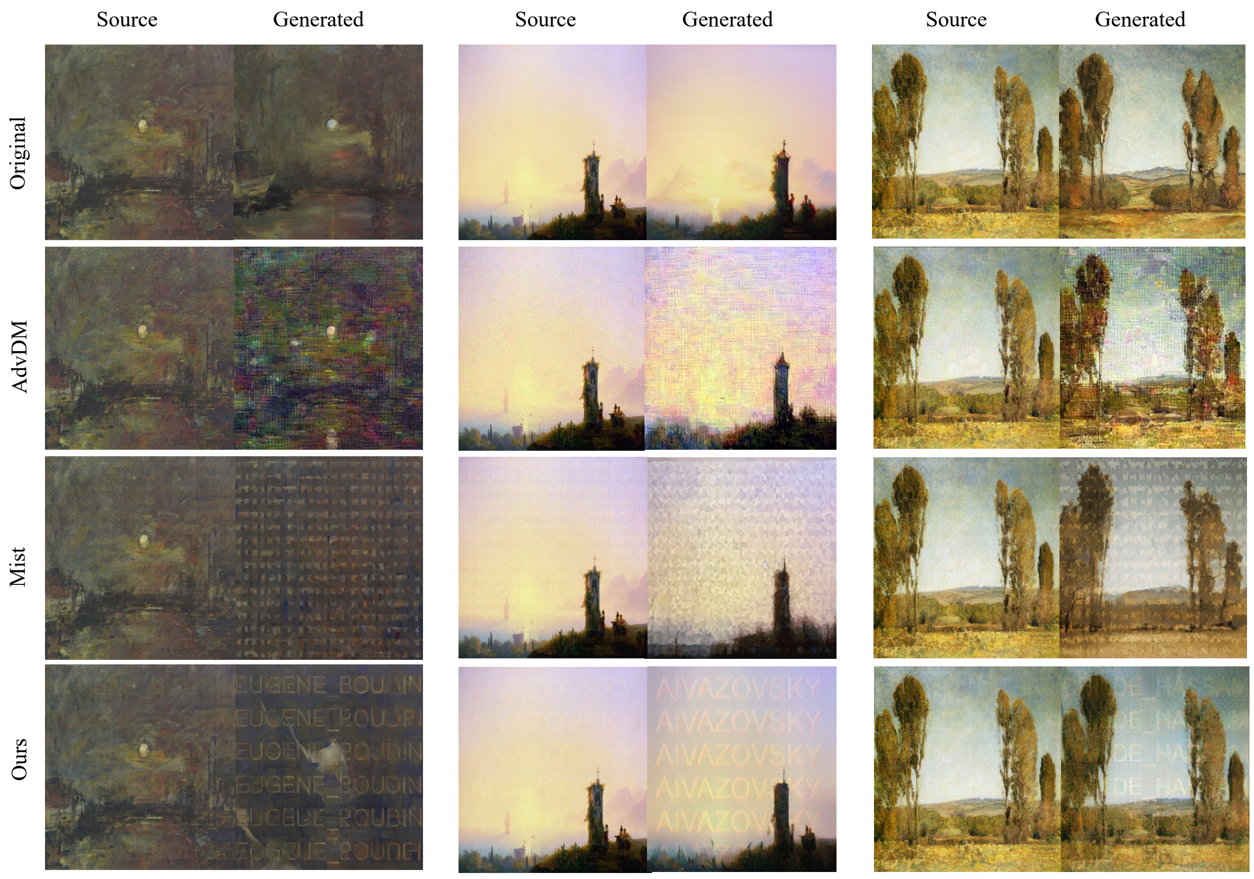

Figure 11. More examples of text-guided image-to-image generation on WikiArt.

E. Settings of Other Generative Models

Stable Diffusion 1.5 (SD 1.5) [6] is a latent text-to-image diffusion model capable of generating realistic images. It can also be applied to image-to-image generation by passing a text prompt and an initial image to condition the generation of new images. We use the text prompt "A painting" for the WikiArt dataset, and "A photo" for the ImageNet dataset. We set the sampling methods to DPM++ 2M Karras, the sampling steps to 50, and the strength to 0.30 as default. Please note that this is the model we used to generate our adversarial examples.

Dreamshaper 8 [7] is a fine-tuned version of Stable Diffusion that addresses some of its limitations. It improves the convergence speed, handles high-dimensional data, and is more robust to noise. We conduct the experiments under image-to-image generation and set the sampling method to DPM++ 2M Karras, the steps to 40, the guidance to 10, and the strength to 0.30 as default.

NovelAI [8] is a service that creates unique stories with virtual companionship. It also has the potential to raise concerns about copyright violations. We conduct experiments under the image-to-image generation scenario in NovelAI. We use the text prompt "A painting" for the WikiArt dataset, and "A photo" for the ImageNet dataset. We set the resolution to 512, the sampling method to DPM++ 2M, the steps to 40, the guidance to 10, and the strength to 0.30 as default.

Runway AI magic tools [9] provides various creative tools to ideate, generate and edit images and videos. We conduct experiments under image variation in this tool. Since there is no parameter that can be adjusted, we directly input our images into this tool and obtain the generated image.

[6]https://huggingface.co/runwayml/stable-diffusion-v1-5
[7]https://civitai.com/models/4384/dreamshaper
[8]https://novelai.net/
[9]https://runwayml.com/ai-magic-tools/image-to-image/

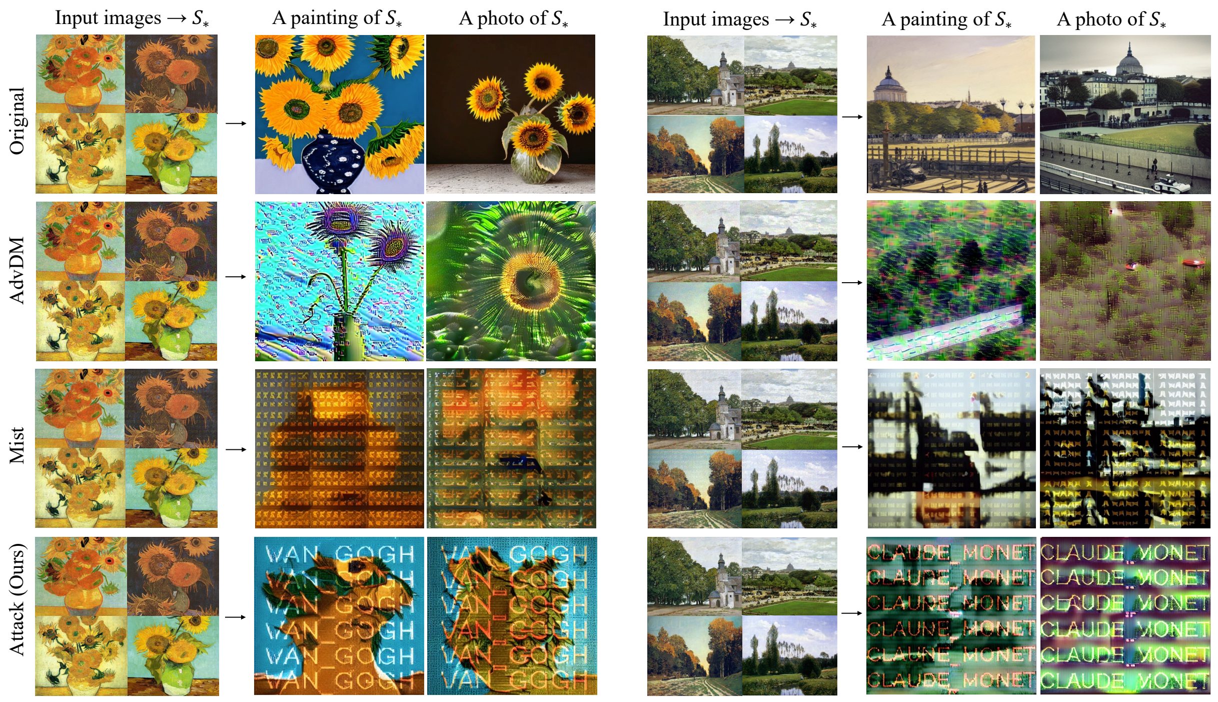

Figure 12. Comparison between our method and the previous methods under textual inversion on WikiArt.

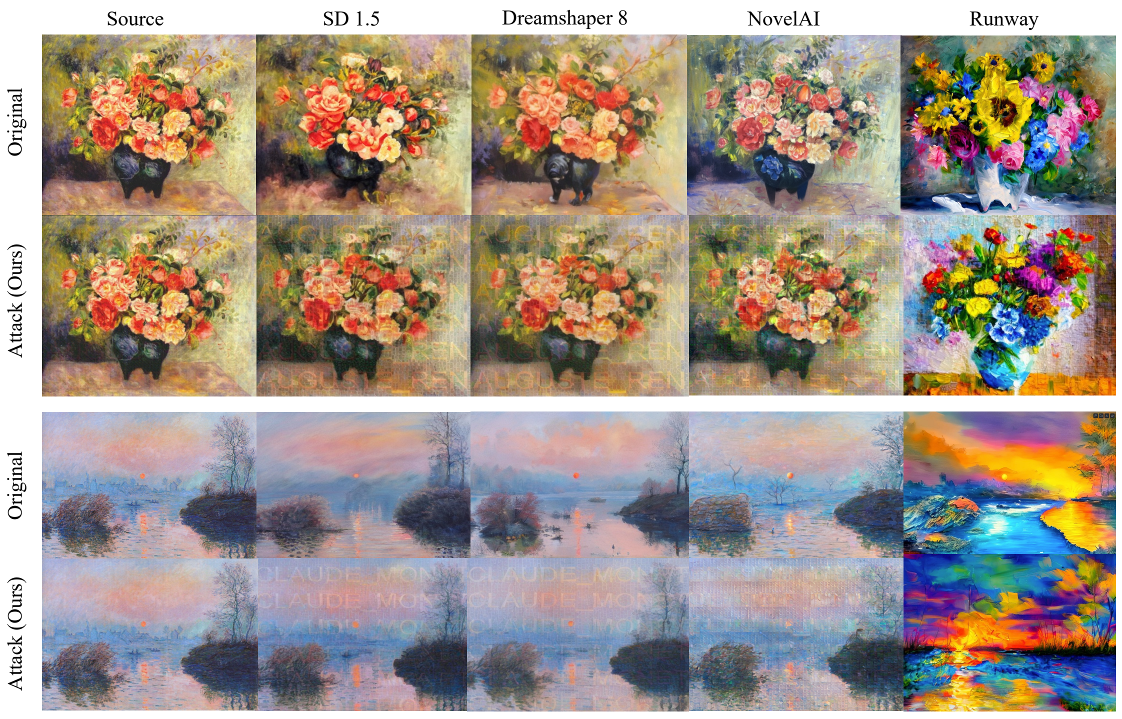

Figure 13. More examples of black-box attack under image-to-image generation using various models or tools.