DiffusionCLIP: Text-Guided Diffusion Models for Robust Image Manipulation (Supplementary Material)

Gwanghyun Kim[1] Taesung Kwon[1] Jong Chul Ye[2,1]
Dept. of Bio and Brain Engineering[1], Kim Jaechul Graduate School of AI[2]
Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea
{gwang.kim, star.kwon, jong.ye}@kaist.ac.kr

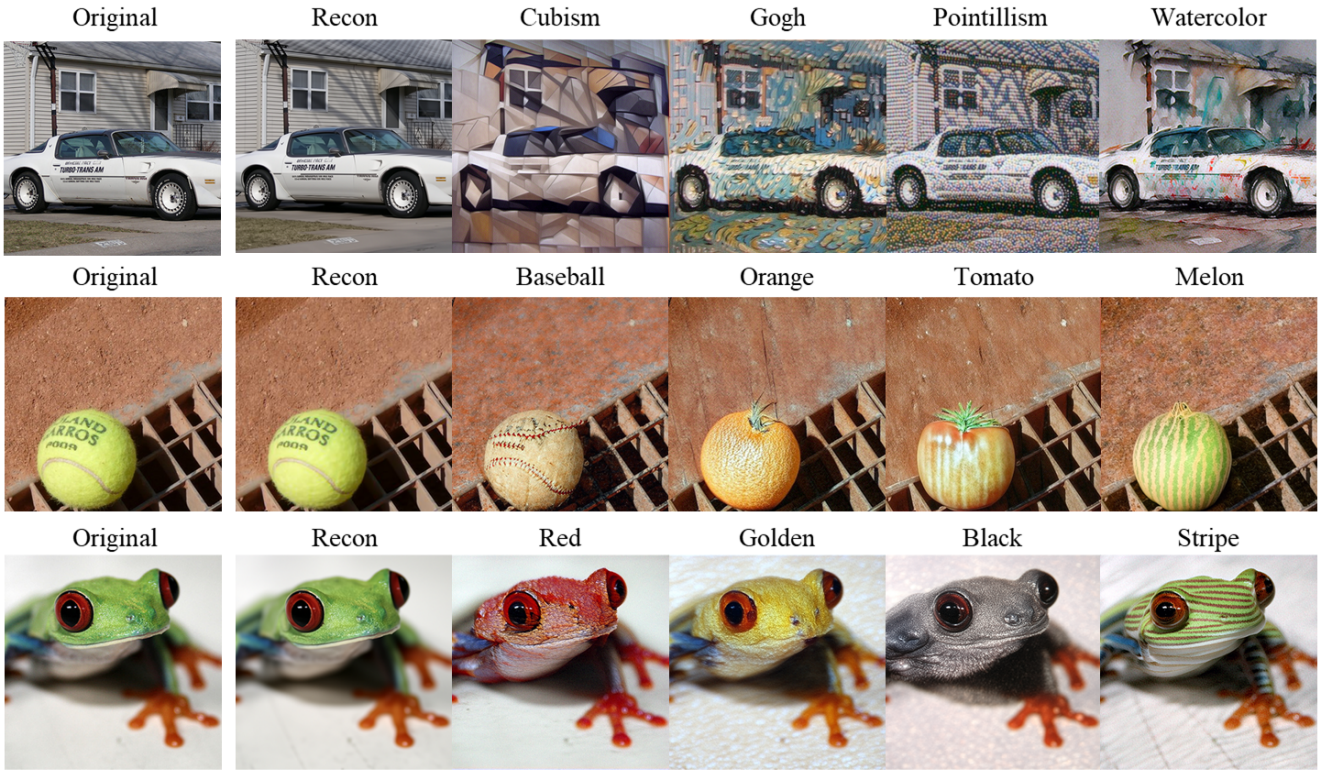

Figure 1. DiffusionCLIP can even perform manipulation of 512×512 images using the ImageNet [37] pretrained diffusion models. Thanks to the near-perfect inversion capability, DiffusionCLIP enables the zero-shot text-driven manipulation, moving a step forward to the general text-driven manipulation. In contrast, due to the diversity of the images in ImageNet, GAN-based inversion and its manipulation in the latent space of ImageNet shows limited performance [5, 10]. Hence, zero-shot text-driven manipulation using ImageNet pretrained GAN have been rarely explored. For more results, see Fig. 7, 17, 18 and 19.

A. Details on Related Works

A.1. DDPM, DDIM and ODE Approximation

Denoising diffusion probabalistic models (DDPM). Diffusion probabilistic models [18] are a class of latent variable models based on forward and reverse processes. Suppose that our model distribution $p_\theta(\boldsymbol{x}_0)$ tries to approximate a data distribution $q(\boldsymbol{x}_0)$. Let $\mathcal{X}$ denote the sample space for $\boldsymbol{x}_0$ generated from a sequence of latent variables $\boldsymbol{x}_t$ for $t = 1, \cdots, T$, where $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In the forward process, noises are gradually added to data $\boldsymbol{x}_0$ and the latent sequence set $\boldsymbol{x}_{1:T}$ are generated through the following Markov chain upon a variance schedule defined by $\{\beta_t\}_{t=1}^T$:

$$q(\boldsymbol{x}_{1:T}) := \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}), \tag{1}$$

where

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) := \mathcal{N}(\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I}). \tag{2}$$

Then, $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ can be represented in a closed form as $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_0, (1-\alpha_t)\mathbf{I})$, where $\alpha_t := 1-\beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t}(1-\beta_s)$. Then, we can sample $\boldsymbol{x}_t$ as:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{w}, \text{ where } \boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{3}$$

In the reverse process, $\boldsymbol{x}_T$ is denoised to generate $\boldsymbol{x}_0$ through the following Markov process:

$$p_\theta(\boldsymbol{x}_{0:T}) := p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \tag{4}$$

where $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)\mathbf{I}), \tag{5}$$

where $\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)$ is set to be learnable to improve the sample quality [28] and

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right). \tag{6}$$

and the neural network $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is trained with the following improved objective [18]:

$$\mathcal{L}_{\text{simple}} := \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{w}, t}||\boldsymbol{w} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{w}, t)||_2^2, \tag{7}$$

where $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Denoising diffusion implicit models (DDIM). An alternative non-Markovian forward process that has the same forward marginals as DDPM and corresponding sampling process is proposed in [40]. Here, the forward diffusion is described by

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{z},$$

while the reverse diffusion can be represented as following:

$$\boldsymbol{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) + \sigma_t^2\boldsymbol{z}, \tag{8}$$

where $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\boldsymbol{f}_\theta(\boldsymbol{x}_t, t)$ is a the prediction of $\boldsymbol{x}_0$ at t given $\boldsymbol{x}_t$:

$$\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) := \frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)}{\sqrt{\bar{\alpha}_t}}, \tag{9}$$

and $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is computed by (7).

This sampling allows using different reverse samplers by changing the variance of the reverse noise σ_t. Especially, by setting this noise to 0, which is a DDIM sampling process [40], the sampling process becomes deterministic, enabling to conversation latent variables into the data consistently and to sample with fewer steps.

ODE approximation. In fact, DDIM can be considered as a Euler method to solve ODE. Specifically, Eq. (8) can be represented as:

$$\sqrt{\frac{1}{\bar{\alpha}_{t-1}}}\boldsymbol{x}_{t-1} - \sqrt{\frac{1}{\bar{\alpha}_t}}\boldsymbol{x}_t = \left(\sqrt{\frac{1}{\bar{\alpha}_{t-1}}-1} - \sqrt{\frac{1}{\bar{\alpha}_t}-1}\right)\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) \tag{10}$$

If we set $\boldsymbol{y}_t := \sqrt{1/\bar{\alpha}_t}\boldsymbol{x}_t$ and $p_t := \sqrt{1/\bar{\alpha}_t - 1}$, we can rewrite Eq. (10) as follows:

$$\boldsymbol{y}_{t-1} - \boldsymbol{y}_t = (p_{t-1} - p_t)\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t). \tag{11}$$

In the limit of small steps, this equation goes to ODE

$$dy_t = \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)dp_t.$$

Then, the reversal of this ODE can be derived as follows:

$$\boldsymbol{y}_{t+1} - \boldsymbol{y}_t = (p_{t+1} - p_t)\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t), \tag{12}$$

which becomes:

$$\sqrt{\frac{1}{\bar{\alpha}_{t+1}}}\boldsymbol{x}_{t+1} - \sqrt{\frac{1}{\bar{\alpha}_t}}\boldsymbol{x}_t = \left(\sqrt{\frac{1}{\bar{\alpha}_{t+1}}-1} - \sqrt{\frac{1}{\bar{\alpha}_t}-1}\right)\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t). \tag{13}$$

Finally, the above equation can be written as:

$$\boldsymbol{x}_{t+1} = \sqrt{\bar{\alpha}_{t+1}}\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) + \sqrt{1-\bar{\alpha}_{t+1}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t), \tag{14}$$

which is equal to our forward DDIM process formulation that is used in Sec. 3.2.

A.2. Additional Related Works

Diffusion-based image manipulation. Recent diffusion models have demonstrated impressive performance in image generation [13, 18, 21, 39–42] with additional advantages of great mode coverage and stable training.

Despite this recent progress, only a few studies [8, 26] have been carried out for image manipulation with diffusion models, such as local editing and the image translation from unseen domain to the trained domain. In ILVR [8], image translation where the low-frequency component of the reference image is conditioned at each transition during the sampling process is introduced. In SDEdit [26], images with the user's local edit or strokes are first noised via the stochastic SDE process, and subsequently denoised by simulating the reverse SDE to generate the realistic image in the pretrained domain. However, it is not clear how these methods can be extended for more general image manipulation applications, such as attribute manipulation, translation from the trained domain to multiple unseen domains, etc.

On the other hand, DiffusionCLIP enables text-guided image manipulation with an infinite number of types of text-driven attributes, and translation of images in the pretrained or an unseen domain to another unseen domain.

GAN-based image manipulation. Image manipulation methods have been mostly implemented using GAN models. Conditional GAN methods [7, 11, 20, 29, 31, 46, 56, 57] learn direct mappings from original images to target images. However, these methods need additional training and collection of the dataset with a huge amount of manual effort whenever the new controls are necessary.

In GAN inversion based methods [1–4, 6, 17, 34, 35, 43, 45, 48, 54, 55], an input image is first converted to a latent vector so that the image can be manipulated by modifying the latent or fine-tuning the generator. In recent works [16, 30], GAN inversion is combined with the CLIP loss [32], so that image manipulation given simple text prompts can be achieved without additional training dataset for target distribution.

However, image manipulation by GAN inversion still demands further investigation, because many datasets are still hard to invert due to the limited inversion capability of GAN models [19, 23, 34]. Even the encoder-based GAN inversion approaches [3, 34, 43], which is the current state-of-the-art (SOTA) methods, often fail to reconstruct images with novel poses, views, and details, inducing the unintended change in the manipulation results. This issue becomes even worse in the case of images from a dataset with high variances such as church images in LSUN Church [50] or ImageNet dataset [37].

On the other hand, DiffusionCLIP allows near-perfect inversions, so that it can perform zero-shot text-driven image manipulation successfully, preserving important details even for images from a dataset with high variance. We can even translate the image from an unseen domain into another unseen domain or generate images in an unseen domain from the strokes. In the following, we illustrate the detailed procedure with pseudocode.

B. Details on Methods

B.1. DiffusionCLIP Fine-tuning

We adopt a two-step approach as detailed in Algorithm 1. First, real images or sampled images from pretrained domain $\{\boldsymbol{x}_0^{(i)}\}_{i=1}^N$ are inverted to the latents $\{\boldsymbol{x}_{t_0}^{(i)}\}_{i=1}^N$ via deterministic forward DDIM processes [40] with the pretrained diffusion model $\boldsymbol{\epsilon}_\theta$. To accelerate the process, instead of performing forward diffusion until the last time step T, we use fewer discretization steps $\{\tau_s\}_{s=1}^{S_{\text{for}}}$ such that $\tau_1 = 0, \tau_{S_{\text{for}}} = t_0$.

In the second step, we start to update $\boldsymbol{\epsilon}_{\hat{\theta}}$, a copy of the pretrained diffusion model. For each latent in $\{\boldsymbol{x}_{t_0}^{(i)}\}_{i=1}^N$, the image is sampled through the reverse DDIM process [40] and the model is updated guided by CLIP loss $\mathcal{L}_{\text{direction}}$ and identity loss $\mathcal{L}_{\text{ID}}$ to generate images that represent y_{tar}. The second step is repeated K times until converged.

GPU-efficient fine-tuning. During the fine-tuning, the model is updated by back-propagating the gradient from the last step as illustrated in Fig. 2(a) and Algorithm 2. Although this method shows great manipulation performance, as the gradient pass the model S_{gen} times, the GPU usage can

Algorithm 1: DiffuisonCLIP fine-tuning

Input: $\boldsymbol{\epsilon}_\theta$ (pretrained model), $\{\boldsymbol{x}_0^{(i)}\}_{i=1}^N$ (images to precompute), y_{ref} (reference text), y_{tar} (target text), t_0 (return step), S_{for} (# of inversion steps), S_{gen}(# of generation steps), K (# of fine-tuning iterations)

Output: $\boldsymbol{\epsilon}_{\hat{\theta}}$ (fine-tuned model)

// Step 1: Precompute latents

Define $\{\tau_s\}_{s=1}^{S_{\text{for}}}$ s.t. $\tau_1 = 0, \tau_{S_{\text{for}}} = t_0$.
for $i = 1, 2, \dots, N$ **do**
  for $s = 1, 2, \dots, S_{\text{for}} - 1$ **do**
    $\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\tau_s}^{(i)}, \tau_s)$; $\boldsymbol{f} \leftarrow \boldsymbol{f}_\theta(\boldsymbol{x}_{\tau_s}^{(i)}, \tau_s)$
    $\boldsymbol{x}_{\tau_{s+1}}^{(i)} \leftarrow \sqrt{\alpha_{\tau_{s+1}}}\boldsymbol{f} + \sqrt{1-\alpha_{\tau_{s+1}}}\boldsymbol{\epsilon}$
  Save the latent $\boldsymbol{x}_{t_0}^{(i)} = \boldsymbol{x}_{\tau_{S_{\text{for}}}}^{(i)}$.

// Step 2: Update the diffusion model

Clone the pretrained model $\boldsymbol{\epsilon}_{\hat{\theta}} \leftarrow \boldsymbol{\epsilon}_\theta$
Define $\{\tau_s\}_{s=1}^{S_{\text{gen}}}$ s.t. $\tau_1 = 0, \tau_{S_{\text{gen}}} = t_0$.
for $k = 1, 2, \dots, K$ **do**
  for $i = 1, 2, \dots, N$ **do**
    Clone the latent $\hat{\boldsymbol{x}}_{t_0}^{(i)} \leftarrow \boldsymbol{x}_{t_0}^{(i)}$.
    for $s = S_{\text{gen}}, S_{\text{gen}} - 1, \dots, 2$ **do**
      $\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon}_{\hat{\theta}}(\boldsymbol{x}_{\tau_s}^{(i)}, \tau_s)$; $\boldsymbol{f} \leftarrow \boldsymbol{f}_{\hat{\theta}}(\boldsymbol{x}_{\tau_s}^{(i)}, \tau_s)$
      $\boldsymbol{x}_{\tau_{s-1}}^{(i)} \leftarrow \sqrt{\alpha_{\tau_{s-1}}}\boldsymbol{f} + \sqrt{1-\alpha_{\tau_{s-1}}}\boldsymbol{\epsilon}$
    $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{direction}}(\hat{\boldsymbol{x}}_0^{(i)}, y_{\text{tar}}; \boldsymbol{x}_0^{(i)}, y_{\text{ref}})$
    $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{total}} + \mathcal{L}_{\text{id}}(\hat{\boldsymbol{x}}_0^{(i)}, \boldsymbol{x}_0^{(i)})$
    Take a gradient step on $\nabla_{\hat{\theta}}\mathcal{L}_{\text{total}}$.

Algorithm 2: GPU-efficient fine-tuning

// Step 2: Update the diffusion model

Clone the pretrained model $\boldsymbol{\epsilon}_{\hat{\theta}} \leftarrow \boldsymbol{\epsilon}_\theta$
Define $\{\tau_s\}_{s=1}^{S_{\text{gen}}}$ s.t. $\tau_1 = 0, \tau_{S_{\text{gen}}} = t_0$.
for $k = 1, 2, \dots, K$ **do**
  for $i = 1, 2, \dots, N$ **do**
    Clone the latent $\hat{\boldsymbol{x}}_{t_0}^{(i)} \leftarrow \boldsymbol{x}_{t_0}^{(i)}$.
    for $s = S_{\text{gen}}, S_{\text{gen}} - 1, \dots, 2$ **do**
      $\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon}_{\hat{\theta}}(\boldsymbol{x}_{\tau_s}^{(i)}, \tau_s)$; $\boldsymbol{f} \leftarrow \boldsymbol{f}_{\hat{\theta}}(\boldsymbol{x}_{\tau_s}^{(i)}, \tau_s)$
      $\boldsymbol{x}_{\tau_{s-1}}^{(i)} \leftarrow \sqrt{\alpha_{\tau_{s-1}}}\boldsymbol{f} + \sqrt{1-\alpha_{\tau_{s-1}}}\boldsymbol{\epsilon}$
    $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{direction}}(\boldsymbol{f}, y_{\text{tar}}; \boldsymbol{x}_0^{(i)}, y_{\text{ref}})$
    $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{total}} + \mathcal{L}_{\text{id}}(\boldsymbol{f}, \boldsymbol{x}_0^{(i)})$
    Take a gradient step on $\nabla_{\hat{\theta}}\mathcal{L}_{\text{total}}$.

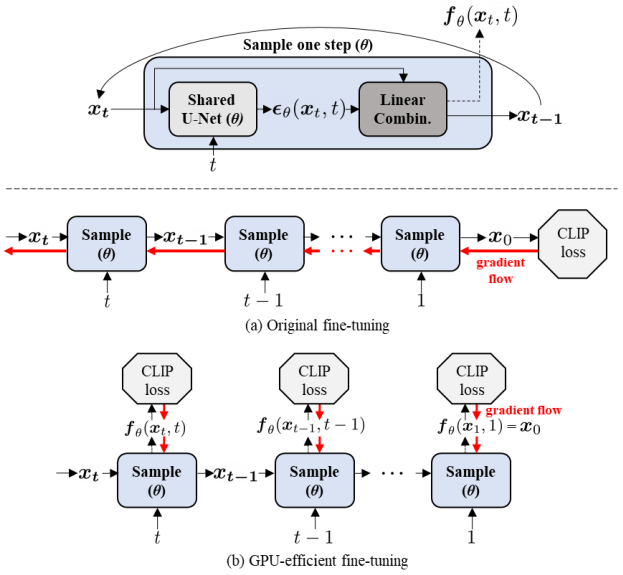

Figure 2. Gradient flows during the fine-tuning and GPU-efficient fine-tuning. In GPU-efficient fine-tuning, loss calculation and a gradient step are proceeded at each time step t with $\boldsymbol{f}_\theta(\boldsymbol{x}_t, t)$, the prediction of $\boldsymbol{x}_0$ at t.

be burdensome. Therefore, we additionally propose GPU-efficient fine-tuning method. Here, as shown in in Fig. 2(b), the back-propagation from the loss functions is performed at each time step t. GPU-efficient fine-tuning can require half of VRAM usage compared to the original fine-tuning, but it requires twice as much time due to calculating loss and making steps at each time step. More details of running time can be found in Sec. G. We show the result of manipulating ImageNet [37] 512×512 images using GPU-efficient fine-tuning method in Fig. 7, 17, 18 and 19.

Image manipulation via fine-tuned model. Once the diffusion model $\boldsymbol{\epsilon}_{\hat{\theta}}$ is fine-tuned for the target control y_{tar}, the manipulation process of a input image $\boldsymbol{x}_0$ is quite simple as in Algorithm 3. Specifically, $\boldsymbol{x}_0$ is inverted to $\boldsymbol{x}_{t_0}$ through the forward DDIM process with the original pretrained model $\boldsymbol{\epsilon}_\theta$, followed by the reverse DDIM process with the fine-tuned model $\boldsymbol{\epsilon}_{\hat{\theta}}$ resulting $\hat{\boldsymbol{x}}_0$. We use the same t_0 as used in the fine-tuning.

B.2. Image Translation between Unseen Domains

By combining the method in SDEdit [26] and the manipulation with the fine-tuned model by DiffusionCLIP as detailed in Algorithm 4, we can even translate an image from an unseen domain into another unseen domain. In the first step, the input image in the source unseen domain $\boldsymbol{x}_0$ is first perturbed to $\boldsymbol{x}'_{t_0}$ through the stochastic forward DDPM process [18] until the return step t_0. Next, the image in the pretrained domain $\boldsymbol{x}'_0$ is sampled through the reverse DDIM process with the original pretrained model $\boldsymbol{\epsilon}_\theta$. These

Algorithm 3: DiffuisonCLIP manipulation

Input: $\boldsymbol{x}_0$ (input image), $\boldsymbol{\epsilon}_{\hat{\theta}}$ (fine-tuned model), $\boldsymbol{\epsilon}_\theta$ (pretrained model), t_0 (return step), S_{for} (# of inversion steps), S_{gen}(# of generation steps)

Function `Manipulation`$(\boldsymbol{x}_0, \boldsymbol{\epsilon}_{\hat{\theta}}, *)$**:**
  Define $\{\tau_s\}_{s=1}^{S_{\text{for}}}$ s.t. $\tau_1 = 0, \tau_{S_{\text{for}}} = t_0$.
  for $s = 1, 2, \ldots, S_{\text{for}} - 1$ **do**
    $\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\tau_s}, \tau_s)$; $\boldsymbol{f} \leftarrow \boldsymbol{f}_\theta(\boldsymbol{x}_{\tau_s}, \tau_s)$
    $\boldsymbol{x}_{\tau_{s+1}} \leftarrow \sqrt{\alpha_{\tau_{s+1}}}\boldsymbol{f} + \sqrt{1 - \alpha_{\tau_{s+1}}}\boldsymbol{\epsilon}$
  Define $\{\tau_s\}_{s=1}^{S_{\text{gen}}}$ s.t. $\tau_1 = 0, \tau_{S_{\text{gen}}} = t_0$
  $\hat{\boldsymbol{x}}_{t_0} \leftarrow \boldsymbol{x}_{t_0}$
  for $s = S_{\text{gen}}, S_{\text{gen}} - 1, \ldots, 2$ **do**
    $\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon}_{\hat{\theta}}(\hat{\boldsymbol{x}}_{\tau_s}, \tau_s)$; $\boldsymbol{f} \leftarrow \boldsymbol{f}_{\hat{\theta}}(\hat{\boldsymbol{x}}_{\tau_s}, \tau_s)$
    $\hat{\boldsymbol{x}}_{\tau_{s-1}} \leftarrow \sqrt{\alpha_{\tau_{s-1}}}\boldsymbol{f} + \sqrt{1 - \alpha_{\tau_{s-1}}}\boldsymbol{\epsilon}$
  return $\hat{\boldsymbol{x}}_0$ (manipulated image)

forward-generative processes are repeated for K_{DDPM} times until the image $\boldsymbol{x}'_0$ is close to the image in the pretrained domain.

In the second step, $\boldsymbol{x}'_0$ is manipulated into the image $\hat{\boldsymbol{x}}_0$ in the CLIP-guided unseen domain with the fine-tuned model $\boldsymbol{\epsilon}_{\hat{\theta}}$ as in Algorithm 3.

Algorithm 4: Translation between unseen domains

Input: $\boldsymbol{x}_0$ (image in an unseen domain or stroke), $\boldsymbol{\epsilon}_{\hat{\theta}}$ (fine-tuned model), K_{DDPM} (# of iterations of Step 1), $\boldsymbol{\epsilon}_\theta$ (pretrained model), t_0 (return step), S_{for} (# of inversion steps), S_{gen}(# of generation steps)

Output: $\hat{\boldsymbol{x}}_0$ (manipulated image)

`// Step 1: Source unseen → Pretrained`

Define $\{\tau_s\}_{s=1}^{S_{\text{gen}}}$ s.t. $\tau_1 = 0, \tau_{S_{\text{gen}}} = t_0$.
$\boldsymbol{x}'_0 \leftarrow \boldsymbol{x}_0$
for $k = 1, 2, \ldots, K_{\text{DDPM}}$ **do**
  $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\boldsymbol{x}'_{t_0} \leftarrow \sqrt{\alpha_{t_0}}\boldsymbol{x}'_0 + (1 - \alpha_{t_0})\boldsymbol{w}$
  for $s = S_{\text{gen}}, S_{\text{gen}} - 1, \ldots, 2$ **do**
    $\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{x}'_{\tau_s}, \tau_s)$; $\boldsymbol{f} \leftarrow \boldsymbol{f}_\theta(\boldsymbol{x}'_{\tau_s}, \tau_s)$
    $\boldsymbol{x}'_{\tau_{s-1}} \leftarrow \sqrt{\alpha_{\tau_{s-1}}}\boldsymbol{f} + \sqrt{1 - \alpha_{\tau_{s-1}}}\boldsymbol{\epsilon}$

`// Step 2: Pretrained → Target unseen`

$\hat{\boldsymbol{x}}_0 \leftarrow$ `Manipulation`$(\boldsymbol{x}'_0, \boldsymbol{\epsilon}_{\hat{\theta}}, *)$

B.3. Noise Combination

With the multiple diffusion models fine-tuned for the different controls $\{\boldsymbol{\epsilon}_{\hat{\theta}_i}\}_{i=1}^M$, we can change multiple attributes through only one sampling process. Specifically, we can flexibly mix several single attribute fine-tuned models with

different combinations as described in Algorithm 5, without having to fine-tune new models with target texts that define multiple attributes.

More specifically, we first invert an input image $\boldsymbol{x}_0$ into $\boldsymbol{x}_{t_0}$ via the forward DDIM process with the original pretrained diffusion model $\boldsymbol{\epsilon}_\theta$ as single attribute manipulation. Then, we use the multiple fine-tuned models during the reverse DDIM process. By applying different time dependent weight $\gamma_i(t)$ satisfying $\sum_{i=1}^{M}\gamma_i(t)=1$ for each model, we can control the degree of change for multiple attributes. Of note, we can also apply this noise combination method for controlling the degree of change during single attribute manipulation. By mixing the noise from the original pretrained model $\boldsymbol{\epsilon}_\theta$ and the fine-tuned model $\boldsymbol{\epsilon}_{\hat{\theta}}$ concerning a single γ, we can perform interpolation between the original image and the manipulated image smoothly.

Algorithm 5: Multi-attribute transfer

Input: $\boldsymbol{x}_0$ (input image), $\{\boldsymbol{\epsilon}_{\hat{\theta}_i}\}_{i=1}^{M}$ (multiple fine-tuned models), $\boldsymbol{\epsilon}_\theta$ (pretrained model), $\{\gamma(t)_i\}_{i=1}^{M}$ (sequence of model weights), t_0 (return step), S_{for} (# of inversion steps), S_{gen}(# of generation steps)

Output: $\hat{\boldsymbol{x}}_0$ (manipulated image)

Define $\{\tau_s\}_{s=1}^{S_{\text{for}}}$ s.t. $\tau_1=0, \tau_{S_{\text{for}}}=t_0$.
for $s=1,2,\dots,S_{\text{for}}-1$ **do**
  $\boldsymbol{\epsilon}\leftarrow\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{\tau_s},\tau_s)$; $\boldsymbol{f}\leftarrow\boldsymbol{f}_\theta(\boldsymbol{x}_{\tau_s},\tau_s)$
  $\boldsymbol{x}_{\tau_{s+1}}\leftarrow\sqrt{\alpha_{\tau_{s+1}}}\boldsymbol{f}+\sqrt{1-\alpha_{\tau_{s+1}}}\boldsymbol{\epsilon}$
Define $\{\tau_s\}_{s=1}^{S_{\text{gen}}}$ s.t. $\tau_1=0, \tau_{S_{\text{gen}}}=t_0$.
$\hat{\boldsymbol{x}}_{t_0}\leftarrow\boldsymbol{x}_{t_0}$
for $s=S_{\text{gen}},S_{\text{gen}}-1,\dots,2$ **do**
  $\boldsymbol{\epsilon}\leftarrow\sum_{i=1}^{M}\gamma_i(\tau_s)\boldsymbol{\epsilon}_{\hat{\theta}_i}(\boldsymbol{x}_{\tau_s},\tau_s)$
  $\boldsymbol{f}\leftarrow\sum_{i=1}^{M}\gamma_i(\tau_s)\boldsymbol{f}_{\hat{\theta}_i}(\boldsymbol{x}_{\tau_s},\tau_s)$
  $\hat{\boldsymbol{x}}_{\tau_{s-1}}\leftarrow\sqrt{\alpha_{\tau_{s-1}}}\boldsymbol{f}+\sqrt{1-\alpha_{\tau_{s-1}}}\boldsymbol{\epsilon}$

C. Details on Network

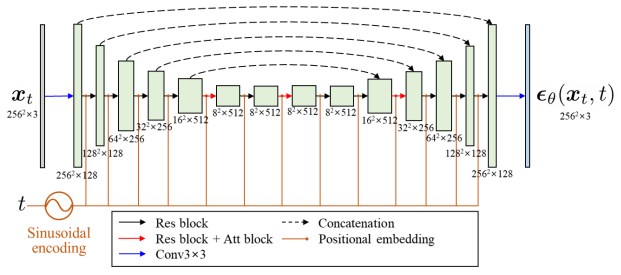

Figure 3. The shared U-Net architecture across t of the diffusion model that generates 256×256 images. The model receives $\boldsymbol{x}_t$ and t as inputs and outputs $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t)$.

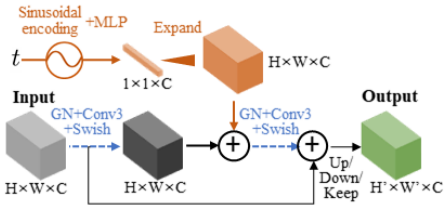

Figure 4. Details of Res block.

Most of existing diffusion models receives $\boldsymbol{x}_t$ and t as inputs to the network $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t)$. We use the DDPM [18] models pre-trained on 256×256 images in CelebA-HQ [22], LSUN-Bedroom and LSUN-Church [50] datasets. This model adopts the U-Net [36] architecture based on Wide-ResNet [51] shared across t as represented in Fig. 3. In specific, the model is composed of the encoder part, middle part, decoder part, and time embedding part. In the encoder part, the 8×8 feature is generated from the 256×256 input image via 1 input convolution and 5 Res blocks. One Res block is composed of two convolutional blocks including Group normalization [47] and Swish activation [33] with the residual connection as in Fig. 4. At the 16×16 resolution, self-attention blocks are added to the Res block. The middle part consists of 3 Res blocks and the second block includes a self-attention block. In the decoder part, the output whose resolution is the same as the input is produced from the feature after the middle part through 5 Res blocks and 1 output convolution with skip connections from the features in the encoder part. In the time embedding part, the diffusion time t is embedded into each Res blocks as represented in Fig. 4 after the Transformer sinusoidal encoding as proposed in [44]. We use the models pretrained on Celeba-HQ, LSUN-Bedroom, and LSUN-Church models that are used in [26].

For the manipulation of dog faces, we use the improved DDPM [28] models pre-trained on AFHQ-Dog [9]. The architecture is almost same except that the model produces the extra outputs at the output convolution to predict the variance $\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t,t)$ as well as the mean $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t,t)$ which can be predicted from $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t)$. We use the models pretrained on AFHQ-Dog that is used in [8].

For the manipulation of 512×512 images from ImageNet dataset [37], we use the improved DDPM [28] pretrained model that is used in [13]. Different from 256×256 resolution models, self-attention blocks are added to the Res block at the resolution of 8×8, 16×16 and 32×32 resolution.

D. Details and More Results of Comparison

D.1. Reconstruction

Here, we provide details on the quantitative comparison of reconstruction performance between our diffusion-based inversion and SOTA GAN inversion methods, which results are presented in Sec 4.1 and Tab. 1 of our main text.

Baseline models. We use optimization approach [1], pixel2style2pixel (pSp) encoder [34], Encoder for Editing (e4e) [43], ReStyle encoder [3] and HFGI encoder [45] as our baseline models. pSp encoder adopts a Feature Pyramid Network and [25] inverts the image into $\mathcal{W}+$ space of StyleGAN. In contrast, e4e converts the image to the latent in $\mathcal{W}$ space, which enables to explain the trade-offs between distortion and editing quality. Restyle encoder is a residual-based encoder, improving its performance using iterative refinement. HFGI encoder further improves the inversion performance leveraging the adaptive distortion alignment module and the distortion consultation module.

Comparison setting. We followed the experimental settings as described in [45]. We invert the first 1,500 CelebA-HQ images. Then, we measure the quality of reconstruction from the inverted latent using MAE, LPIPS, SSIM metrics. All results except the result of our method are from the [45]. For our method, we set $(S_{\text{for}}, S_{\text{gen}})$ to (200, 40), which is our general setting.

D.2. Human Evaluation

Comparison setting. We conduct user study to evaluate real face image manipulation performance on CelebA-HQ [22] with our method, StyleCLIP global direction (GD) [30] and StyleGAN-NADA [16]. We get 6,000 votes from 50 people using a survey platform. We use the first 20 images in CelebA-HQ testset as general cases and use another 20 images with novel views, hand pose, and fine details as hard cases. For a fair comparison, we use 4 in-domain attributes (angry, makeup, beard, tanned) and 2 out-of-domain attributes (zombie, sketch), which are used in the studies of baselines. Here, we use official pretrained checkpoints and implementation for each approach. We ask the respondents to rank the models by how well the image is manipulated, representing the property of the target attribute and preserving important semantics of the objects.

Results used for evaluation. We provide manipulation results by our method, StyleCLIP-GD and StyleGAN-NADA, which are used for human evaluation, in Fig. 14, 15.

D.3. Quantitative Evaluation

Quality metrics. We use the following quality metrics for quantitative evaluation: Directional CLIP similarity ($\mathcal{S}_{\text{dir}}$), segmentation-consistency (SC), and face identity similarity (ID). Specifically, $\mathcal{S}_{\text{dir}}$ is defined as follows:

$$\mathcal{S}_{\text{dir}}\left(\boldsymbol{x}_{\text{gen}}, y_{\text{tar}}; \boldsymbol{x}_{\text{ref}}, y_{\text{ref}}\right) := \frac{\langle \Delta I, \Delta T \rangle}{\|\Delta I\| \|\Delta T\|}, \tag{15}$$

where

$$\Delta T = E_T(y_{\text{tar}}) - E_T(y_{\text{ref}}), \Delta I = E_I(\boldsymbol{x}_{\text{gen}}) - E_I(\boldsymbol{x}_{\text{ref}}).$$

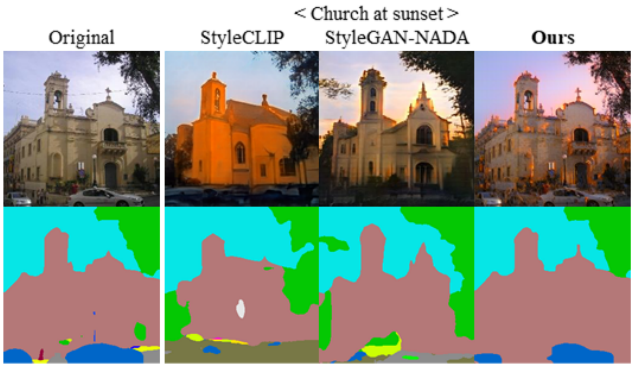

Figure 5. Example of segmentation results from the manipulation results by different methods.

Here, E_I and E_T are CLIP's image and text encoders, respectively, and y_{tar}, $\boldsymbol{x}_{\text{gen}}$ are the text description of a target and the generated image, respectively. Also, y_{ref}, $\boldsymbol{x}_{\text{ref}}$ denote the source domain text and image, respectively. Next, SC is a pixel accuracy when the segmentation result from $\boldsymbol{x}_{\text{ref}}$ by the pretrained segmentation model is set as the label and the result from $\boldsymbol{x}_{\text{gen}}$ is set as the prediction, as shown in Figure 5. Lastly, ID $:= L_{\text{face}}(\boldsymbol{x}_{\text{gen}}, \boldsymbol{x}_{\text{ref}})$ where $\mathcal{L}_{\text{face}}$ is the face identity loss in [12].

Our goal is to achieve the better score in terms of $\mathcal{S}_{\text{dir}}$, SC, and ID to demonstrate high attribute-correspondence ($\mathcal{S}_{\text{dir}}$) as well as well-preservation of identities without unintended changes (SC, ID).

Comparison setting. To compute $\mathcal{S}_{\text{dir}}$, we use a pretrained CLIP [32]. To calculate SC, we use pretrained face parsing network [49] and semantic segmentation networks [52, 53]. To compute ID, we use a pretrained face recognition [12] model. Then, we performed comparison with StyleCLIP [30] and StyleGAN-NADA [16]. We use 1,000 test images from CelebA-HQ [22] and LSUN-Church [50], respectively. We use the manipulation results for three attributes in CelebA-HQ (makeup, tanned, gray hair) and LSUN-Church (golden, red brick, sunset). These attributes are required to confirm that the manipulation results correspond to the target text without the changes of identities and shapes of the source objects.

D.4. Comparison of Church Image Manipulation

We additionally provide the manipulation of 256×256 church images from LSUN-Church [50] with StyleCLIP latent optimization (LO) [30] and StyleGAN-NADA [16] in Fig. 16.

D.5. Diffusion-based Manipulations

We compare our model fine-tuning method with latent optimization and conditional sampling method [13] guided by CLIP loss.

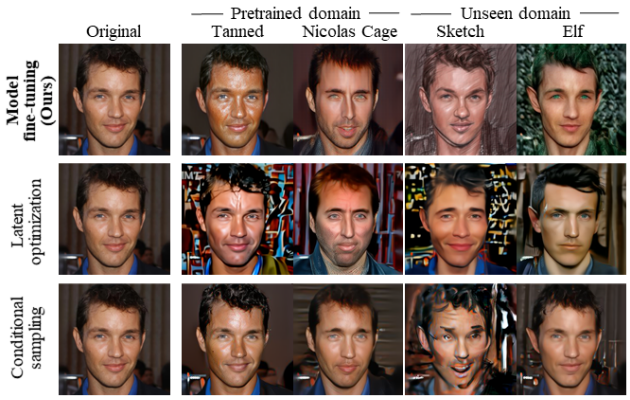

Figure 6. Comparison between diffusion-based manipulation methods.

For the latent optimization of the diffusion models, we use the same objective (Eq. (10) in the main manuscript) as the model fine-tuning. However, we optimize the inverted latent $\hat{\boldsymbol{x}}_{t_0}$ instead of the model $\epsilon_{\hat{\theta}}$. For conditional sampling, the sampling process is guided by the gradient of CLIP loss with respect to the latent as a classifier guides the process in [13]. This method requires a noisy classifier that can classify the image with noise, but the noisy CLIP model is not publicly available and its training will be too expensive. To mitigate this issue, we use the method proposed by [27]. Instead of using noisy CLIP, they use the gradient from the normal CLIP loss with the predicted $\boldsymbol{x}_0$ given $\boldsymbol{x}_t$, which we denoted as $\boldsymbol{f}_\theta(\boldsymbol{x}_t, t)$ in Eq. (9) at every step.

In Fig. 6, we display a series of the real image manipulation given the text prompt by our model fine-tuning method, latent optimization and conditional sampling. We can see that the manipulation results via latent optimization and conditional sampling methods failed to manipulate the images to the unseen domain. The reason is that the manipulation using latent optimization and conditional sampling is restricted by the learned distribution of the pretrained model. On the other hand, the proposed model fine-tuning method shows superior manipulation performance.

D.6. Other GAN Baselines

Comparison with VQGAN-CLIP. VQGAN-CLIP [15, 32] recently show the impressive results of CLIP-guided conditional generation of artistic images. It also provides the style transfer, which optimizes the latent from the input image guided by CLIP loss. We compare DiffusionCLIP with VQGAN-CLIP for the manipulation of 512×512 images from ImageNet [37]. We follow official implementation for VQGAN-CLIP. For our method, we utilize GPU-efficient fine-tuning method with the diffusion model pretrained on 512×512 ImageNet which is used in [13]. We set $(S_{\text{for}}, S_{\text{gen}}) = (40, 12)$. In the first two rows of Fig. 7, our method successfully translates the image into target style,

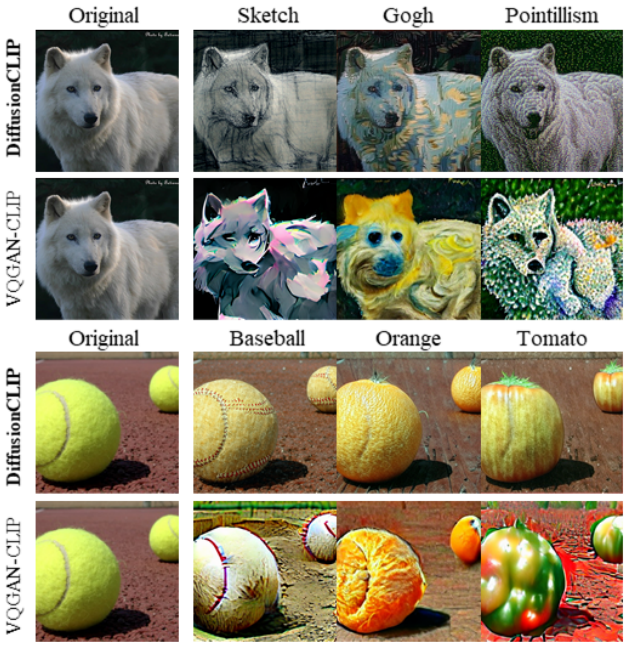

Figure 7. Comparison with VQGAN-CLIP [15,32] using 512×512 images from ImageNet [37]

preserving the identity of the object. However, the manipulation results by VQGAN-CLIP do not show representative properties of target styles. In the bottom two rows of Fig. 7, our method shows excellent semantic manipulation results preserving the details of backgrounds, while the results from VQGAN-CLIP show severe unintended changes.

Figure 8. Comparision with other GAN inversion-based manipulation: StyleSpace [48] and InterfaceGAN [38].

Other GAN inversion-based manipulation. We also compare our method with non text-driven manipulation methods based on GAN inversion: StyleSpace [48] and InterfaceGAN [38]. StyleSpace manipulates the latent inverted by e4e [43] in StyleGAN2 [24] $\mathcal{W}+$ space. InterfaceGAN manipulates the latent inverted by IDInvert [54] in StyleGAN [23] $\mathcal{W}+$ space. As shown in Fig. 8, StyleSpace and

InterfaceGAN fail to manipulate the images with hand poses, suggesting practical limitations. However, our method successfully manipulates the images without artifacts.

E. Additional Results

Manipulation of 512×512 images from ImageNet. Here, we provide the results of the manipulation of 512×512 images from ImageNet [37]. We leverage GPU-efficient fine-tuning with the diffusion model pretrained on 512×512 ImageNet which is used in [13]. We set $(S_{\text{for}}, S_{\text{gen}}) = (40, 12)$. We set $(S_{\text{for}}, S_{\text{gen}}) = (40, 12)$ and other hyperparameters are equally applied as manipulation of 256×256 images. We first show the style transfer results of general images in Fig. 17. We show text-driven semantic manipulation results from tennis ball into other objects in Fig. 18. Finally, we show the manipulation of frog images in Fig. 19.

Image translation between unseen domains. In Fig. 20 we display additional results of image translation between unseen domains, where animation images, portrait art, and strokes are translated into Pixar, paintings by Gogh and Neanderthal men. Note that we do not require any curated dataset for both source and target domain.

Figure 9. Failure cases.

Failure cases. Due to the dependency on the performance of CLIP encoder, DiffusionCLIP sometimes fails to manipulate images as shown in Fig. 9. For example, it is difficult to manipulate human face images into objects such as computers, chairs, pencils. Also, manipulation to target controls that happen or become famous recently may fail because their representations are not reflected inside CLIP encoders.

F. Hyperparameter and Ablation Study

F.1. Dependency on $S_{\text{for}}, S_{\text{gen}}$ and t_0

In Table 1, the reconstruction from the latents through the inversion process on face images are evaluated using MAE, LPIPS and SSIM. As S_{for} and S_{gen} increase, the reconstruction quality increases. However, in case that $S_{\text{for}} < S_{\text{gen}}$, the quality stays in the similar degree or even decreases, causing the artifacts as the cases of $(S_{\text{for}}, S_{\text{gen}}) = (6, 40)$ and $(S_{\text{for}}, S_{\text{gen}}) = (200, 6)$ in Fig. 10 in the main manuscript. When $(S_{\text{for}}, S_{\text{gen}})$ is fixed, as the return step t_0 increases, the quality decreased because the intervals between the steps become larger.

F.2. Identity Loss

Figure 10. Ablation study of identity loss.

Here, we analyze the importance of identity loss. We use ℓ_1 loss as the identity loss, and in the case of human face image manipulation, the face identity loss in [12] is used. Whether to use these identity losses is determined by the target control. We show the examples in Fig. 10. If preserving the identity of the human face is important for the target control such as 'Makeup', it is recommended to use face identity loss as we can see in the first row in Fig. 10. ℓ_1 can help further preserve the background details. If the target control doesn't require the exact identity preserving as artistic transfer as the second rows of Fig. 10, the identity loss can hinder the change. The examples of usage of hyperparameters depending on the target text prompts are represented in Table 2.

F.3. Dependency on Fine-tuning Epochs K

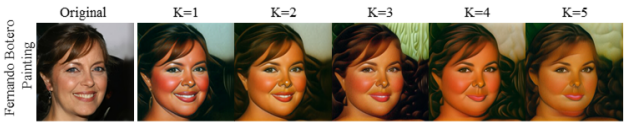

Figure 11. Changes according to the fine-tuning epochs.

To fine-tune diffusion models, we use Adam optimizer with an initial learning rate of 4e-6 which is increased linearly by 1.2 per 50 iterations. Hence, as we can see in the example of changes are represented in Fig. 11, the images generated from the fine-tuned models change closer to the target control as the epoch K increases.

F.4. Dependency on the Number of Precomputed Images N

As we mentioned before, if several latents have been precomputed, we can further reduce the time for fine-tuning by recycling the latent to synthesize other attributes. In this case, the number of precomputed images N is a hyperparameter to be controlled. We test the cases with different

Table 1. Quantitative analysis on reconstruction quality with respect to S_{for}, S_{gen} and t_0.

t_0	S_{for}	S_{gen}	MAE ↓	LPIPS ↓	SSIM ↑
300	6	6	0.047	0.185	0.732
		40	0.061	0.221	0.704
		200	0.063	0.224	0.694
	40	6	0.027	0.110	0.863
		40	0.023	0.091	0.891
		200	0.023	0.086	0.895
	200	6	0.024	0.095	0.885
		40	0.020	0.073	0.914
		200	0.019	0.065	0.923
400	6	6	0.055	0.208	0.673
		40	0.073	0.255	0.655
		200	0.077	0.260	0.643
	40	6	0.031	0.128	0.827
		40	0.025	0.100	0.880
		200	0.024	0.093	0.885
	200	6	0.028	0.108	0.862
		40	0.024	0.076	0.910
		200	0.020	0.068	0.919

t_0	S_{for}	S_{gen}	MAE ↓	LPIPS ↓	SSIM ↑
500	6	6	0.065	0.237	0.602
		40	0.085	0.286	0.615
		200	0.090	0.292	0.602
	40	6	0.037	0.148	0.779
		40	0.027	0.109	0.868
		200	0.026	0.101	0.874
	200	6	0.032	0.126	0.827
		40	0.022	0.082	0.901
		200	0.021	0.073	0.912
600	6	6	0.084	0.283	0.501
		40	0.101	0.325	0.564
		200	0.106	0.330	0.552
	40	6	0.047	0.175	0.706
		40	0.029	0.120	0.852
		200	0.028	0.108	0.862
	200	6	0.041	0.147	0.778
		40	0.024	0.087	0.893
		200	0.022	0.076	0.907

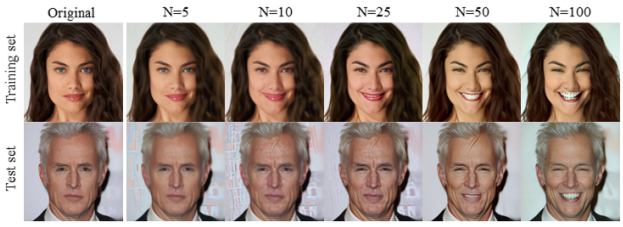

Figure 12. Dependency on the number of precomputed images N

N. We fine-tune the models with $N = 5, 10, 25, 50, 100$, fixing the learning rates to 4e-6 and the number of iterations to 100. We found that as increasing the N, the image can be manipulated more as shown as Fig. 12.

F.5. Stochastic Manipulation

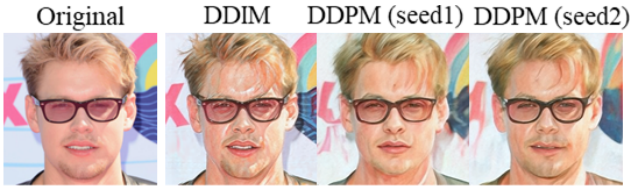

Figure 13. Effect of stochastic manipulation with random seeds.

We analyzed how the results change when stochastic DDPM sampling is used rather than deterministic DDIM sampling. As shown in Figure 13, the images can be modified in many ways, which can be useful for artistic transfer.

F.6. Hyperparameters according to Target Text y_{tar}

We provide examples of hyperparameter settings according to y_{tar} in Table 2. Our method has a similar number of hyperparameters as other text-driven methods such as StyleCLIP [30] and StyleGAN-NADA [16]. In our method, the actual hyperparameters for different controls are just t_0, λ_{L1}, λ_{ID}. These can be chosen simply based on insight as to whether the target requires severe shape changes. The target controls demanding severe changes of shape or color such as change of species or artistic style transfer require high t_0 without no identity losses, while the target controls were preserving the identity of the object is important to require low t_0 and the use of identity losses.

G. Running Time and Resources

Here, we provide the details on the running time of training and inference for each procedure using NVIDIA Quadro RTX 6000 in the case of manipulating 256×256 size images.

DiffuisonCLIP fine-tuning. As illustrated in Sec B.1, DiffusionCLIP fine-tuning process can be split into the latent precomputing procedure and the model updating procedure. The latent precomputing procedure is carried out just once for the same pre-trained diffusion. When we use S_{for} of 40 as normal, the inversion for each image takes 1.644 seconds (all the reported times are the average times of 30 iterations). So, when we precompute the latents from the 50 images, it finished at about 82.2 seconds. For the model updating process, one update step including the generative process, loss calculation, and taking a gradient step takes 0.826 seconds when the batch size is 1 and S_{gen} is 6. So, 1 epoch with 50 precomputed image-latent pairs takes 41.3 seconds. The total epochs K are range from 1 to 10 depending on types of the target text y_{tar}, so the total time for the model updating takes from 41.3 seconds to 7 minutes.

Table 2. Examples of hyperparameter settings according to y_{tar}.

Type	y_{tar}	y_{ref}	t_0	λ_{L1}	λ_{ID}
Human face	*Tanned face*	*face*	300	0.3	0.3
	Face with makeup	*Face*	300	0.3	0.3
	Face without makeup	*Face*	300	0.3	0.3
	Angry face	*face*	500	0.3	0.3
	Person with beards	*Person*	400	0.3	0.3
	Person with curly hair	*Person*	400	0.3	0.3
	Person with red hair	*Person*	500	0.3	0.3
	Person with grey hair	*Person*	500	0.3	0.3
	Old person	*person*	400	0.3	0.3
	Mark Zuckerberg	*Person*	600	0.3	0
	Painting by Gogh	*photo*	600	0	0
	Painting in Modigliani style	*Photo*	600	0	0
	Sketch	*Photo*	600	0.3	0
	3D render in the style of Pixar	*Photo*	600	0.3	0
	Portrait by Firda Kahlo	*Photo*	600	0.3	0
	Super Saiyan	*Human*	600	0	0
	Tolken elf	*Human*	600	0	0
	Zombie	*Human*	600	0	0
	The Jocker	*Human*	600	0	0
	Neanderthal	*Human*	600	0	0
Dog face	*Smiling Dog*	*Dog*	600	0.3	-
	Yorkshire Terrier	*Dog*	600	0	-
	Hamster	*Dog*	600	0	-
	Bear	*Dog*	500	0	-
	Wolf	*Dog*	500	0	-
	Fox	*Dog*	500	0	-
	Nicolas Cage	*Dog*	600	0	-
	Zombie	*Dog*	600	0.3	-
	Venom	*Dog*	600	0.3	-
	Painting by Gogh	*Photo*	500	0.3	-
Church	*Red brick wall church*	*church*	300	0.3	-
	Golden church	*church*	400	0.3	-
	Snow covered church	*church*	500	0.3	-
	Wooden house	*church*	500	0.3	-
	Ancient traditional Asian tower	*church*	500	0.3	-
	Departmetn store	*church*	500	0.3	-
Bedroom	*Blue tone bedroom*	*bedroom*	500	0.3	-
	Green tone bedroom	*bedroom*	500	0.3	-
	Wooden bedroom	*bedroom*	500	0.3	-
	Golden bedroom	*bedroom*	400	0.3	-
	Palace bedroom	*bedroom*	500	0.3	-
	Princess bedroom	*bedroom*	500	0.3	-
	Watercolor art with thick brushstrokes	*Photo*	600	0.3	-

When using GPU-efficient model updating, loss calculation and taking a gradient step takes 1.662 seconds which is almost twice as the original fine-tuning. Therefore, total fine-tuning time will be increased as twice.

The latent precomputing procedure requires about 6GB. The original model and GPU-efficient model updating require 23GB and 12GB of VRAM, respectively.

Manipulation of images from pretrained domain. With the quick manipulation $(S_{\text{for}}, S_{\text{gen}}) = (40, 6)$, it takes 1.644 seconds and 0.314 seconds for the inversion process and the generative process, respectively, resulting in 1.958 seconds total. The quick manipulation still produces great results that can be well used for image manipulation in practice. When we set $(S_{\text{for}}, S_{\text{gen}})$ to (200, 40), it takes 8.448 seconds and 1.684 seconds for the inversion process and the generative process respectively, leading to 10.132 seconds in total. This application and the following applications all require at least 6GB of VRAM.

Image translation between unseen domains. Image translation between unseen domains and stroke-conditioned unseen domain generation requires K_{DDPM} forward DDPM and reverse DDIM process added to one forward and reverse DDIM process. Thanks to the possibility of the sampling $\boldsymbol{x}'_t$ in closed form, the time for forward DDPM and reverse DDIM process can be reduced into the time for the reverse DDIM process 0.314 seconds when $S_{\text{gen}} = 6$. K_{forward} is set to 1-10, so K_{DDPM} forward DDPM and reverse DDIM process takes 0.314-3.14 seconds. When time for one forward and reverse DDIM process is added, the whole process takes 2.272-5.098 seconds with $(S_{\text{for}}, S_{\text{gen}}) = (40, 6)$ and 10.446-13.272 seconds with $(S_{\text{for}}, S_{\text{gen}}) = (200, 40)$.

Multi-attribute transfer. We can change multiple attributes through only one generative process. It takes 2.602 seconds when $(S_{\text{for}}, S_{\text{gen}}) = (40, 6)$ and 14.744 seconds when $(S_{\text{for}}, S_{\text{gen}} = (200, 40)$.

Trade-off between the inference time and preparation time. Latent optimization-based manipulation methods [30] do not require the preparation time for the manipulation. However, they require an optimization process per image. In contrast, our fine-tuning methods, latent mapper in StyleCLIP [30] and StyleGAN-NADA [16] require the set-up for manipulation, which is training the model. However, once the model is fine-tuned, we can apply the model to all images from the same pretrained domain. In terms of training time, our method takes 1-7 minutes, which is faster than the latent mapper of StyleCLIP (10-12hours) and similar to StyleGAN-NADA (a few minutes).

Increasing image size. We found that as the image size is increased from 256×256 to 512×512, the running time for each procedure increased as 4 times, and GPU usage increased as twice.

H. Societal Impacts

DiffusionCLIP enables high-quality manipulation of images for people using simple text prompts without professional artistic skills. However, this manipulation can be used maliciously to confuse people with realistic manipulated results. Therefore, we advise users to make use of our method properly. We also advise you to make use of our method carefully for proper purposes.

In this work, we use two types of pretrained models, CLIP [32] and the diffusion models, to manipulate images without additional manual efforts for new target controls. Image encoder and text encoder of CLIP are trained on 400 million image-text pairs gathered from publicly available

sources on the internet to learn visual concepts from natural language supervision. However, although the size of the training dataset is huge, it is not enough for the models to learn general balanced knowledge. As the authors in [32] acknowledged the potential issues from model biases, manipulation using CLIP can introduce biased results. Diffusion models trained on CelebA-HQ [22], AFHQ-dog [9], LSUN-Bedroom, LSUN-Church [50] and ImageNet [37] used in our models can generate biased results during iterations. Especially, the generative models trained on the CelebA-HQ dataset that is composed of face images of celebrities are founded to produce face images of attractive people who are mostly 20-40 years old [14]. We hope that more research is conducted in direction of generative models and representation learning that resolve the bias issues.

Figure 14. Manipulation of hard cases that are used for human evaluation. Hard cases include 20 images with novel poses, views and details in CelebA-HQ [22]. We compare our method with StyleCLIP global direction method [30] and StyleGAN-NADA [16].

Figure 15. Manipulation of general cases that are used for human evaluation. General cases include the first 20 images in CelebA-HQ testset [22]. We compare our method with StyleCLIP global direction method [30] and StyleGAN-NADA [16].

Figure 16. Qualitative comparison of church image manipulation performance with StyleCLIP global direction method [30] and StyleGAN-NADA [16].

Figure 17. Manipulation of 512×512 images using the ImageNet [37] pretrained diffusion models.

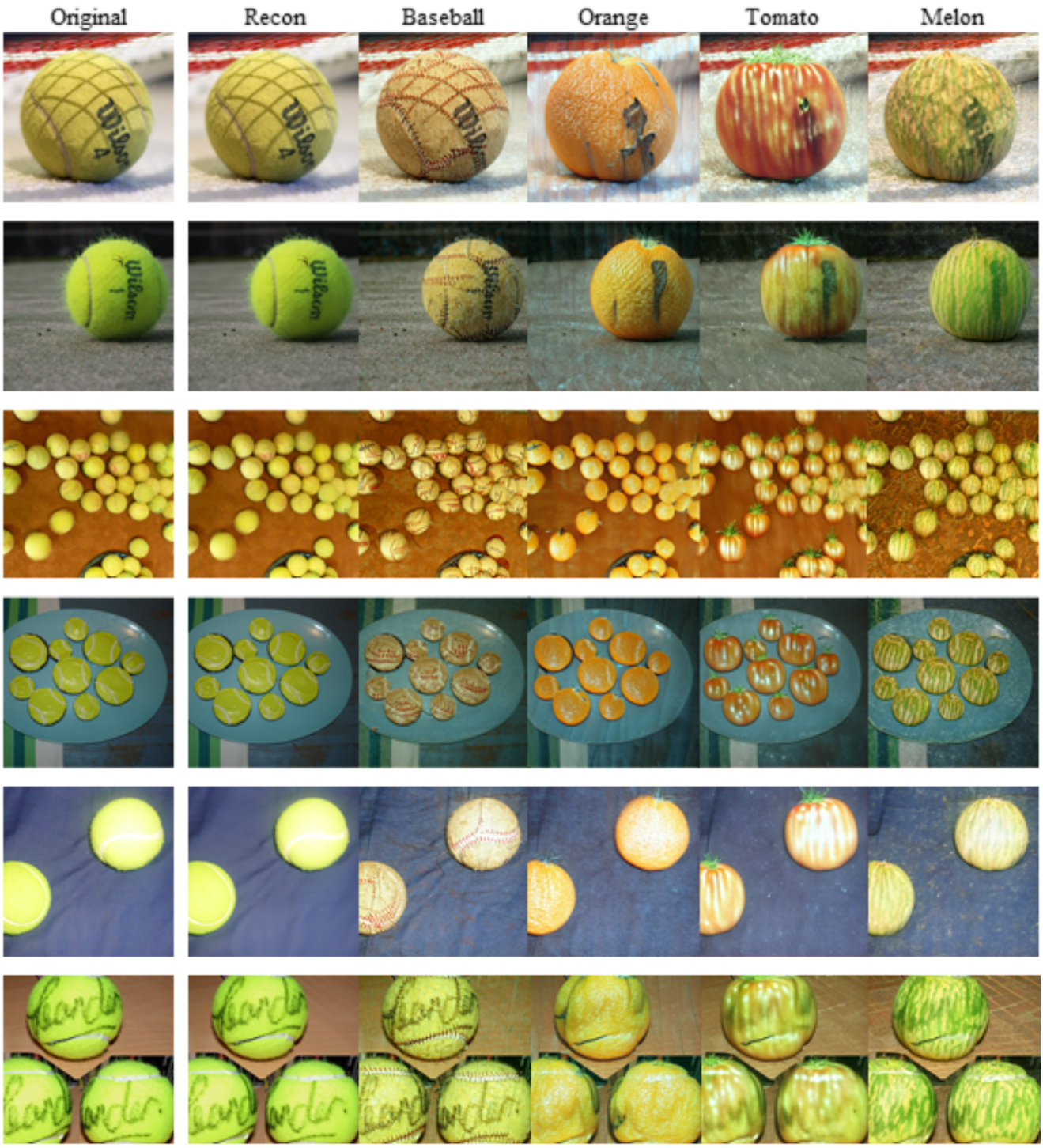

Figure 18. Manipulation of 512×512 images of tennis balls using the ImageNet [37] pretrained diffusion models.

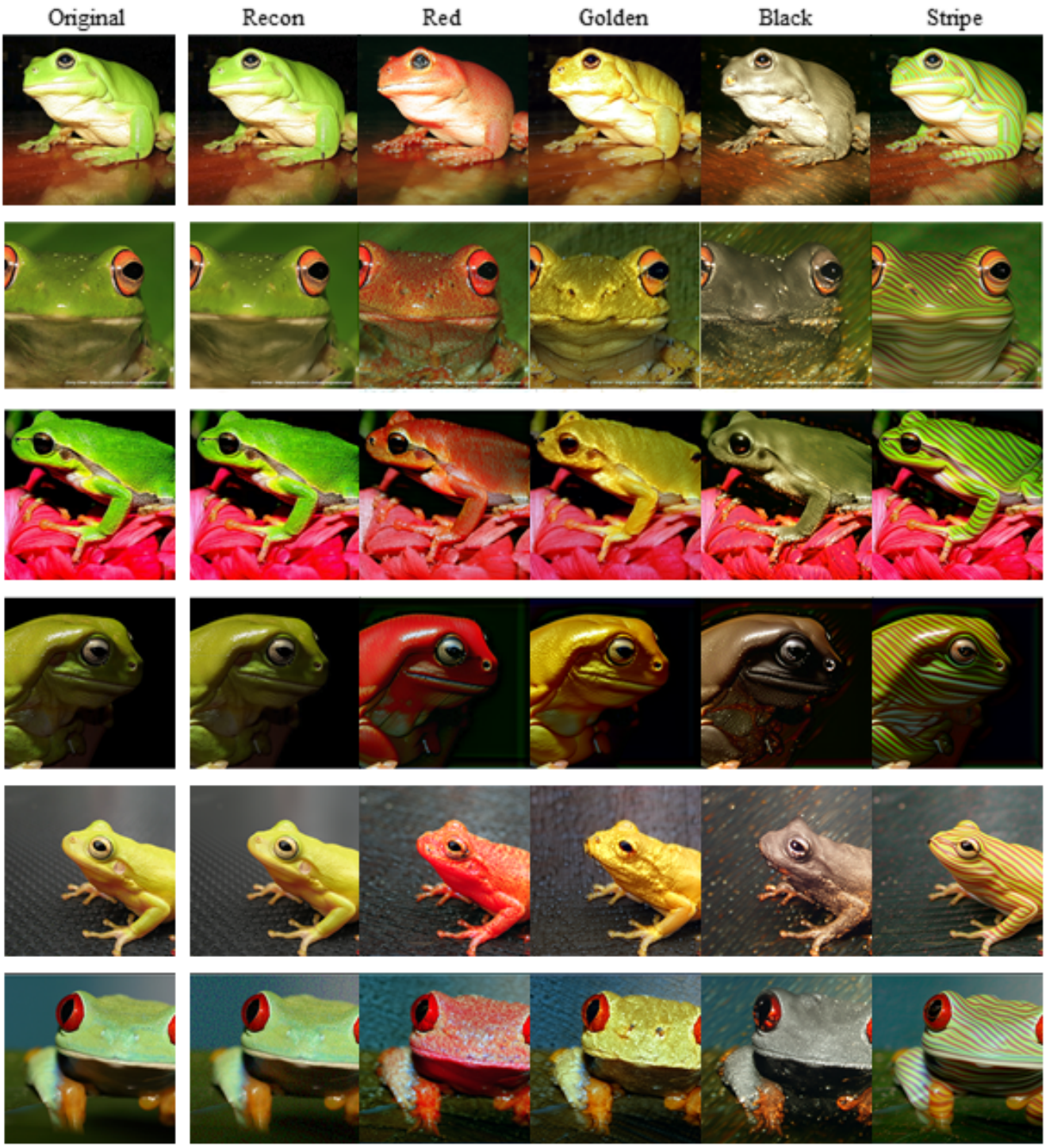

Figure 19. Manipulation of 512×512 images of frogs using the ImageNet [37] pretrained diffusion models.

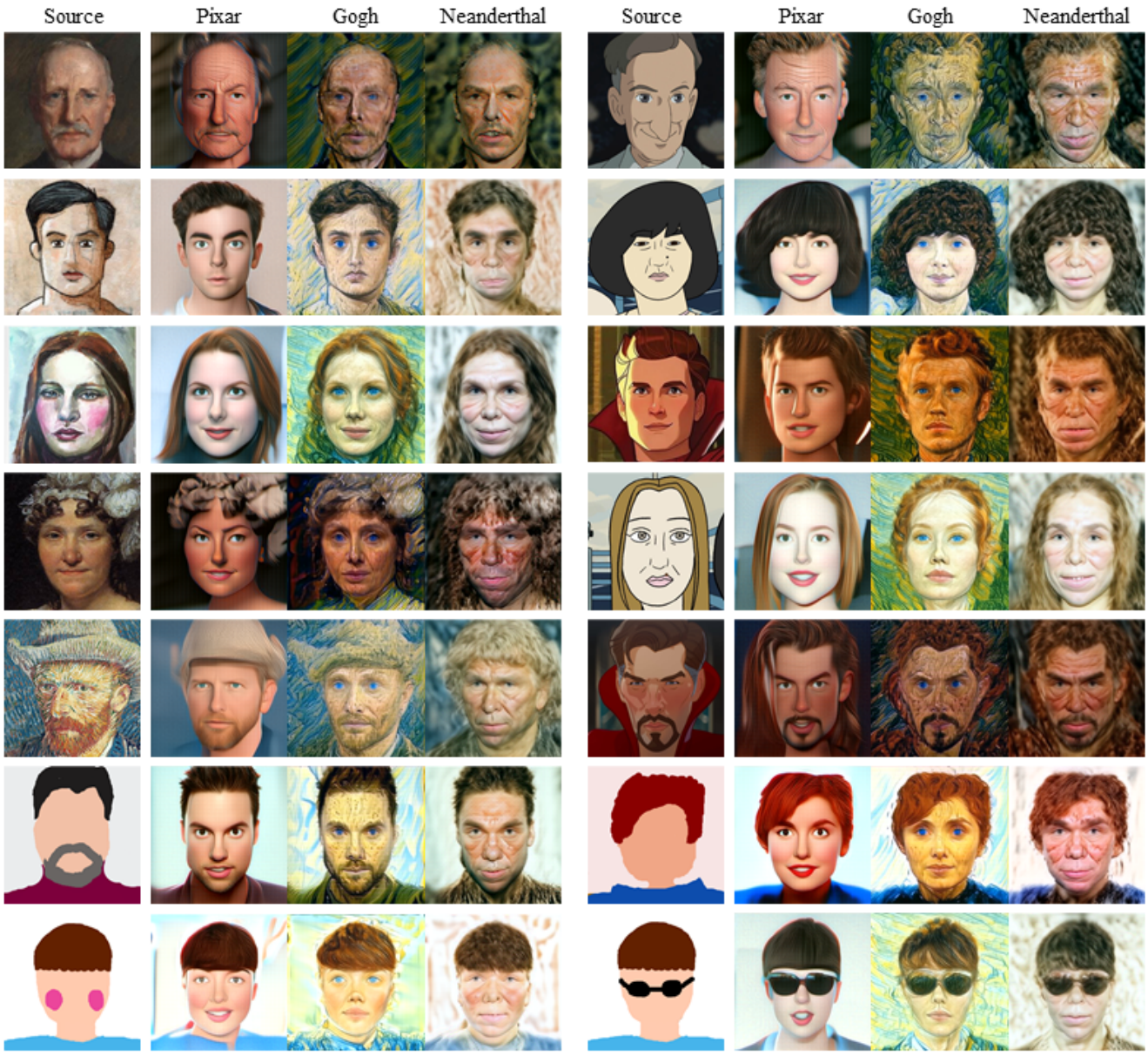

Figure 20. Additional results of image translation between unseen domains.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4432–4441, 2019. 3, 6

[2] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8296–8305, 2020. 3

[3] Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. Restyle a residual-based stylegan encoder via iterative refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021. 3, 6

[4] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *arXiv preprint arXiv:2005.07727*, 2020. 3

[5] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *International Conference on Machine Learning*, pages 537–546. PMLR, 2017. 1

[6] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016. 3

[7] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1511–1520, 2017. 3

[8] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. *arXiv preprint arXiv:2108.02938*, 2021. 2, 5

[9] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 5, 11

[10] Giannis Daras, Augustus Odena, Han Zhang, and Alexandros G Dimakis. Your local gan: Designing two dimensional local attention mechanisms for generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14531–14539, 2020. 1

[11] Tali Dekel, Chuang Gan, Dilip Krishnan, Ce Liu, and William T Freeman. Sparse, smart contours to represent and edit images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3511–3520, 2018. 3

[12] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019. 6, 8

[13] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233*, 2021. 2, 5, 6, 7, 8

[14] Patrick Esser, Robin Rombach, and Björn Ommer. A note on data biases in generative models. *arXiv preprint arXiv:2012.02516*, 2020. 11

[15] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2020. 7

[16] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *arXiv preprint arXiv:2108.00946*, 2021. 3, 6, 9, 10, 12, 13, 14

[17] Jinjin Gu, Yujun Shen, and Bolei Zhou. Image processing using multi-code gan prior. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3012–3021, 2020. 3

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020. 1, 2, 4, 5

[19] Minyoung Huh, Richard Zhang, Jun-Yan Zhu, Sylvain Paris, and Aaron Hertzmann. Transforming and projecting images into class-conditional generative networks. In *European Conference on Computer Vision*, pages 17–34. Springer, 2020. 3

[20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 3

[21] Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Rémi Tachet des Combes, and Ioannis Mitliagkas. Adversarial score matching and improved sampling for image generation. *arXiv preprint arXiv:2009.05475*, 2020. 2

[22] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 5, 6, 11, 12, 13

[23] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. 3, 7

[24] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020. 7

[25] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 6

[26] Chenlin Meng, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021. 2, 4, 5

[27] Clay Mullis and Katherine Crowson. Clip-guided diffusion github repository. In *https://github.com/afiaka87/clip-guided-diffusion*. 7

[28] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*, 2021. 2, 5

[29] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019. 3

[30] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. *arXiv preprint arXiv:2103.17249*, 2021. 3, 6, 9, 10, 12, 13, 14

[31] Tiziano Portenier, Qiyang Hu, Attila Szabo, Siavash Arjomand Bigdeli, Paolo Favaro, and Matthias Zwicker. Faceshop: Deep sketch-based face image editing. *arXiv preprint arXiv:1804.08972*, 2018. 3

[32] 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. *arXiv preprint arXiv:2103.00020*, 2021. 3, 6, 7, 10, 11

[33] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 5

[34] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2287–2296, 2021. 3, 6

[35] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *arXiv preprint arXiv:2106.05744*, 2021. 3

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 5

[37] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1, 3, 4, 5, 7, 8, 11, 15, 16, 17

[38] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 7

[39] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 2

[40] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2, 3

[41] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *arXiv preprint arXiv:1907.05600*, 2019. 2

[42] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 2

[43] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics (TOG)*, 40(4):1–14, 2021. 3, 6, 7

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 5

[45] Tengfei Wang, Yong Zhang, Yanbo Fan, Jue Wang, and Qifeng Chen. High-fidelity gan inversion for image attribute editing. *arXiv preprint arXiv:2109.06590*, 2021. 3, 6

[46] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018. 3

[47] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 5

[48] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12863–12872, 2021. 3, 7

[49] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018. 6

[50] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 3, 5, 6, 11

[51] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 5

[52] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 6

[53] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal on Computer Vision*, 2018. 6

[54] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *European conference on computer vision*, pages 592–608. Springer, 2020. 3, 7

[55] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European conference on computer vision*, pages 597–613. Springer, 2016. 3

[56] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 3

[57] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5104–5113, 2020. 3