TiNO-Edit: Timestep and Noise Optimization for Robust Diffusion-Based Image Editing

Sherry X Chen*[1], Yaron Vaxman[2], Elad Ben Baruch[2], David Asulin[2], Aviad Moreshet[2], Kuo-Chin Lien[3], Misha Sra[1], and Pradeep Sen[1]

[1.]University of California, Santa Barbara [2.]Cloudinary [3.]Layer AI

Abstract

Despite many attempts to leverage pre-trained text-to-image models (T2I) like Stable Diffusion (SD) [25] for controllable image editing, producing good predictable results remains a challenge. Previous approaches have focused on either fine-tuning pre-trained T2I models on specific datasets to generate certain kinds of images (e.g., with a specific object or person), or on optimizing the weights, text prompts, and/or learning features for each input image in an attempt to coax the image generator to produce the desired result. However, these approaches all have shortcomings and fail to produce good results in a predictable and controllable manner. To address this problem, we present TiNO-Edit, an SD-based method that focuses on optimizing the noise patterns and diffusion timesteps during editing, something previously unexplored in the literature. With this simple change, we are able to generate results that both better align with the original images and reflect the desired result. Furthermore, we propose a set of new loss functions that operate in the latent domain of SD, greatly speeding up the optimization when compared to prior losses, which operate in the pixel domain. Our method can be easily applied to variations of SD including Textual Inversion [13] and DreamBooth [27] that encode new concepts and incorporate them into the edited results. We present a host of image-editing capabilities enabled by our approach. Our code is publicly available at https://github.com/SherryXTChen/TiNO-Edit.

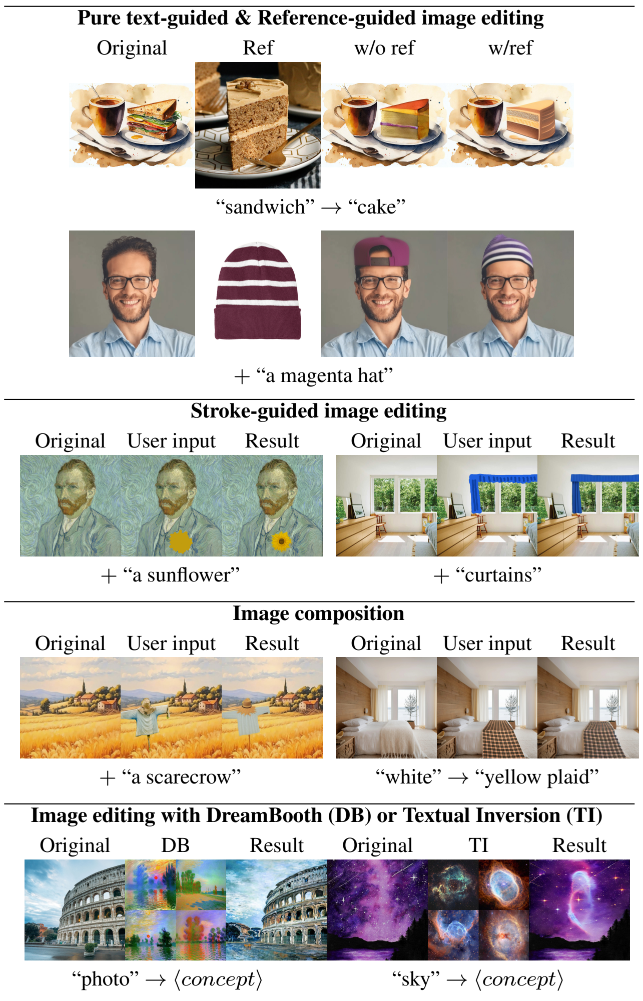


Figure 1. **Overview of capabilities enabled by TiNO-Edit.** TiNO-Edit offers various image-editing capabilities and can be run with DreamBooth (DB) [27] or Textual Inversion (TI) [13]. By leveraging diffusion timestep and noise optimization techniques, it can generate realistic and high quality outputs.

1. Introduction

Computer-generated image synthesis has been studied for decades for its wide range of applications including content creation, marketing and advertising, visualization/simulation, entertainment, and storytelling. Re-

*Corresponding author email: xchen774@ucsb.edu cent advances in diffusion-based text-to-image (T2I) generations [10, 19, 23, 25, 30] models have allowed users to specify, modify, and enhance images in novel ways using text-based prompts or other inputs [35]. Still, these often give undesired results, so the main question is how to use these models in an artistic workflow to enable controllable image editing that can produce the desired results.

To this end, researchers have fine-tuned existing model backbones on new datasets to accommodate specific use cases, including conditioning image editing on instructions [3], using images with inpainting masks [34] or bounding boxes [17], and including other visual information such as edges, segmentation maps, and depth maps [35]. Another line of work has focused on optimizing weights [6, 36], intermediate attention/feature layers [22, 31], or inputs [21, 33] of existing T2I backbones with respect to each image the user wants to edit to try and produce the desired result.

In this last thrust, optimizing text-prompt inputs has gained the most attention. We have seen considerable work on prompt engineering [25] which studies the effect of different (positive or negative) text prompts, including optimizing the text prompt encoding in the CLIP domain [24] as well as adjusting the weights of text prompts [33]. However, as shown in the examples in this paper, these previous approaches still leave a lot to be desired when it comes to high-quality image editing.

Despite previous work that attempts to optimize input parameters to the diffusion model, we observe that there are two other kinds of inputs whose optimization has not been thoroughly explored yet: the noise used to corrupt the input images, and the diffusion timesteps. For noisy images, most work either inverts/reconstructs them with respect to the noise scheduler they use with the diffusion backbones [21, 22, 31, 32] or simply distorts the original image based on a pre-chosen distortion level [1, 8, 20], which is determined by the starting diffusion timestep. Although preliminary research has been conducted to explore how this affects the similarity between the output and the original image [1, 8, 20], these observations have yet to be incorporated into the methods in an automated manner, so it still requires extensive trial-and-error for users to find the optimal settings. Furthermore, the remaining timestep values are fixed based on method hyper-parameters.

Given the important role noisy input images and timesteps play in the diffusion-based image synthesis process, we argue that they should also be optimized based on the image editing objective. To this end, we present TiNO-Edit, an optimization-based method that is built up of one T2I backbone, namely Stable Diffusion (SD) [25], to automatically find the best level of noise that should be added and removed from the original image to achieve the best results. We design a set of loss functions that operate in the latent domain of SD to save computational time and resources. More importantly, TiNO-Edit supports a line of image editing capabilities (Fig. 1), including pure text-guided image editing, image editing guided by reference images, stroke-based image editing, and image composition. It can also be applied with variations of SD including Textual Inversion [13] and DreamBooth [27] that encode new concepts in order to generate outputs with concepts that may otherwise be hard to describe, which is a capability that has been largely overseen.

In summary, our contributions include:

- A novel SD algorithm for image editing that supports various image-editing capabilities.
- A new set of loss functions that save computational time and resources.
- A novel image editing capability of incorporating new concepts from variations of SD

2. Related Work

2.1. Text-to-image diffusion-based models

Diffusion models have shown unprecedented quality in image generation tasks and become the backbone of many text-to-image (T2I) applications [3, 21, 22, 31–36]. These models use text prompts as inputs to the diffusion model to condition the reverse denoising process. Like the general diffusion models, T2I diffusion-based models are able to take Gaussian noise and denoise it using text prompts to generate images that align with those text prompts.

2.2. Text-guided diffusion-based image editing with pre-trained models

T2I diffusion-based models have drawn a lot of attention with their high-quality synthesis results. One line of work takes these pre-trained models as backbones and fine-tunes them to perform specific image editing tasks. For example, ControlNet [35] takes a copy of the U-Net [26] encoder component of SD [25] and attaches it back to the SD backbone to accept visual information including image edge maps, depth maps, segmentation maps, etc. This attached encoder is then fine-tuned to encode that information and guide the generated image output accordingly. Although ControlNet primarily focuses on T2I applications, other work built on it supports text-guided image editing [14].

On the other hand, InstructPix2Pix [3] fine-tunes SD to accept instructions via text prompts to perform image editing. They tailor a dataset of image pairs and corresponding instructions that reflect changes in the image pairs, using the T2I backbone and a large language model GPT-3 [4]. Similarly, Paint by Example [34] is an exemplar-based image editing method via inpainting, where they attach a CLIP-based classification model on SD to accept image examples that will be used to fill the masked region of an input image. The pre-trained SD backbone is then fine-tuned to be conditioned on these examples.

Another approach centers on optimizing weights [6, 36], intermediate attention/feature layers [22, 31], or inputs [21, 33] for each specific image. Similar to the previously mentioned methods, these approaches also need a specific dataset. The dataset may be formed from a single image, such as its patches [36]. The model takes in the noisified patches as well as their positions with respect to the original image and learns to denoise those patches. During inference, both models (before and after fine-tuning) are used to denoise the input. Subsequently, the outputs from each stage are combined through diffusion (reverse denoising) steps to generate the final image.

Intermediate layers of pre-trained models can be used to guide image editing processes. Existing work [11, 22, 31] has looked into various aspects including cross and self-attention maps, as well as activation functions and their effects on object location, shape, size, and appearance in the generated results. Again, image inversion may be applied with respect to the original image [31] to learn relevant information of intermediate layers, which can then be injected in the denoising process when editing this image [31] or be compared with the ones with respect to the edited results and optimize the latter to be aligned with the former [22].

Finally, inputs of T2I models include text prompts, input noise, and timesteps, all of which may be optimized for more desirable editing outcomes. Image editing work that utilizes input optimization usually starts from image inversion (reconstructing images by representing them in the domain of the generative model), which involves finding the reverse denoising path that will lead to the input image [21]. Among all inputs of T2I models, prompts have gained a fair amount of attention as they have a lot of power over the semantics of generative images and, thus are very relevant to image editing results. In the context of SD, prompts can either be positive to describe what the generated images should contain or negative, to describe undesirable visuals that should be excluded from the outputs (e.g. "low quality, jpeg artifacts, ugly..."). Optimization can be performed on both positive [33] or negative prompts [21].

Our optimization-based method falls into the above category. In contrast to prior work, our method maintains both positive and negative prompts while focusing on optimizing input noise and timesteps instead to get the corresponding edits entailed by the prompts.

3. Preliminaries

In this section, we first provide an overview of Diffusion Denoising Implicit Model (DDIM) [30] and Stable Diffusion (SD) [25] algorithms, which our method is based on, followed by an experiment that gives us insight into the effect of various components in them on edited results.

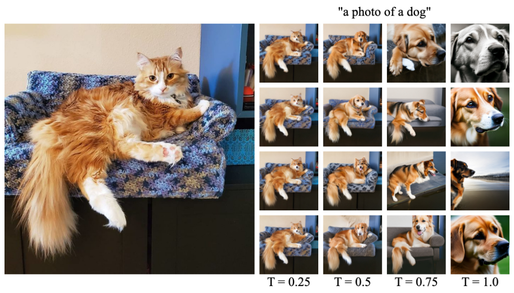


Figure 2. **Effect of starting timestep and noise on image editing.** Suppose we want to change the cat in the left image to a dog, we can input this image and the target prompt "a photo of a dog" to Stable Diffusion (SD) Img2Img [15, 25], along with random Gaussian noise N and a starting time $T \in [0, 1]$ to produce results such as those shown in the grid on the right. Here, we vary T (fixed per column) and N (fixed per row). As T increases, the output matches the target prompt better, but it also diverges more from the original image in terms of composition and pose. Furthermore, different random noise inputs can lead to different visual features.

DDIMs are generative models that produce natural images from noise iteratively. During training, DDIMs model a forward diffusion (FD) process transforming images to noise in S timesteps, denoted as $\{t_k = k \cdot \frac{1}{S}, k \in [0, S]\}$, where the corresponding noisified image I_k per timestep from a pristine image I is

$$I_k = \text{FD}_k(I, N) = \sqrt{\alpha(t_k)} \cdot I + \sqrt{1 - \alpha(t_k)} \cdot N, \quad (1)$$

where noise $N \sim \mathcal{N}(0, \mathbf{I})$ and $\alpha(\cdot)$ is a pre-defined diffusion function. Starting from $\tilde{I}_k = I_k$, DDIMs then model a reverse diffusion (RD) that remove noise iteratively and recover intermediate images as

$$\begin{aligned}\tilde{I}_{k-1} =& \text{RD}_{k,k-1}(\tilde{I}_k, \tilde{N}_k) \\ =& \sqrt{\alpha(t_{k-1})} \left(\frac{\tilde{I}_k - \sqrt{1 - \alpha(t_k)} \cdot \tilde{N}_k}{\sqrt{\alpha(t_k)}} \right) \\ &+ \sqrt{1 - \alpha(t_{k-1})} \cdot \tilde{N}_k,\end{aligned} \quad (2)$$

, where $\tilde{N}_k = \text{UNet}_{\text{DDIM}}(\tilde{I}_k, t_k)$ is the predicted noise from a denoising UNet $\text{UNet}_{\text{DDIM}}$ [26].

The same can be applied to Stable Diffusion (SD) [25] with two key differences. First, SD performs diffusion steps in a latent space of a Variational Auto-Encoder (VAE) [12] with an encoder VAE_{enc} and a decoder VAE_{dec}. The latent representation of I is $L = \text{VAE}_{\text{enc}}(I)$. Secondly, the denoising UNet in SD is conditioned on an additional text prompt p, where $\tilde{N}_k = \text{UNet}_{\text{SD}}(\tilde{L}_k, t_k, p)$, which enables its text-to-image synthesis functionality.

SD Img2Img variant [15] can edit images with text. Given an image I and time $T \in [0, 1]$, I is first noisified with respect to timestep $\max_{t_k \leq T} t_k$ using Eq. 1 and then denoised. The idea behind this, first introduced in

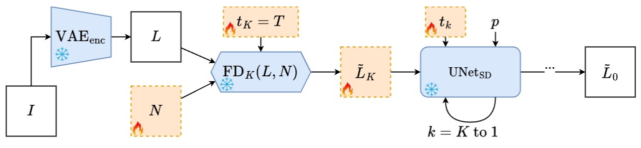


Figure 3. **Optimization parameters.** We find optimization parameters by studying the SD denoising process. The output $\tilde{L}_0$ is only affected by the timesteps t_k ($k \in [1, K]$) and the noisy latent image input $\tilde{L}_k$ for each of the K denoising steps. Note we are assuming that the learning models are all fixed (denoted by the snowflake symbol) and that the number of timesteps K is a constant. $\tilde{L}_k$ can then be traced back through K iterations to the initial latent image input $\tilde{L}_K$ that is computed from starting timestep $t_K = T$ and the Gaussian noise N. Hence, we can achieve our goal by simply optimizing N and time steps t_k for all $k \in [1, K]$.

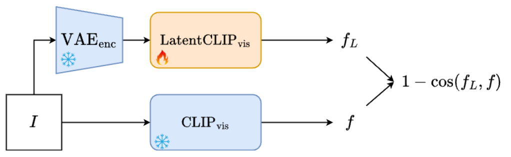
SDEdit [20], is that the less we distort the input, the more likely we are to generate a result similar to it.

However, this raises the question as to what the value of T should be. To give us insight into the problem, we can run a simple experiment, shown in Fig. 2. As we can see, changes in T and N can lead to drastically different results, which motivates our timestep and noise optimization approach explained in the next section.

4. TiNO-Edit

Rather than finding the optimal values manually as in previous approaches, TiNO-Edit aims to automate the process by optimizing the variable parameters related to N and T so that they produce desired results. From the SD denoising process (Fig. 3), we see that the output result $\tilde{L}_0$ is only affected by the timesteps t_k ($k \in [1, K]$) and the input Gaussian noise N, assume all models are fixed and the number of steps K is a constant. We optimize them, starting from $t_k = k \cdot \frac{T}{K}$ and $\mathcal{N}(0, \mathbf{I})$, with respect to our loss functions defined in Sec. 4.2 via gradient descent with the Jacobian from UNet_{SD}. We optimizes all t_k's so they can be non-uniformly spaced and is more flexible than only optimizing T and/or k. The updated t_k's are used in the next optimization step as the timestep inputs to the UNet for the reverse diffusion process. Note that we don't optimize $\alpha(t_k)$ because doing so along with changing t_k will break the dependency between the two learnt by the models and may cause generation quality degrade.

The inputs to our method include the input image I, the target prompt p, the input image description/prompt p_O and additional inputs $\mathcal{A} = \{I_*, M_a, * \in \{r, s, c\}\}$ where a, r, s, c denote the additional image and mask input to the editing capabilities of **a**dding objects, **r**eference-guided image editing, **s**troked-guided image editing, and image **c**omposition if applicable.



VAE_{enc}
$\text{LatentCLIP}_{\text{vis}}$
f_L
$1 - \cos(f_L, f)$
I
CLIP_{vis}
f

Figure 4. **Training LatentCLIP.** Our LatentCLIP visual encoder ($\text{LatentCLIP}_{\text{vis}}$) is a copy of a pre-trained CLIP image encoder (CLIP_{vis}) [24], except the first convolution layer is replaced to accommodate for taking the latent vector $\text{VAE}_{\text{enc}}(I)$[12] as input and output the image feature f_L. The entire LatentCLIP visual encoder is unfrozen (indicated by the fire symbol, as opposed to the snowflake symbol which means the model is frozen) and is trained to minimize the cosine difference between f_L and f, which is the image feature of I from the CLIP image encoder.

4.1. Masking mechanism

During the editing process, it does not make sense to add noise or denoise regions that we want to maintain. So we design a masking mechanism $\text{MSK}(I, p_O, p, \mathcal{A})$ to locate the editing region M from aforementioned inputs. To replace objects in pure text-guided image editing, $M = \text{CLIPSeg}(I, o)$, where $\text{CLIPSeg}(\cdot, \cdot)$ [18] computes regions in I that correspond to objects $\{o : o \in p_O, o \notin p\}$. To perform style transfer, $M_{ij} = 1$ for all pixel locations (i, j). To add objects for pure text-guided or reference-guided image editing, $M = M_a \in \mathcal{A}$. For stroke-guided image editing or image composition, $M_{ij} = 1 - \delta(I_{ij}, (I_*)_{ij})$ using the Kronecker delta function ($\delta(x, y) = 1$ if $x = y$, otherwise 0).

4.2. Optimization loss functions

To achieve the desired output $\tilde{I}_0$ or its latent representation $\tilde{L}_0$, we need to design a set of optimization functions that captures key characteristics the output should have with respect to the original image I or its latent representation L.

First, we want to enforce semantic similarity between the two with a CLIP-based[24] semantic loss function. But instead of operating it in the pixel domain like previous methods [9, 33], we have designed our own CLIP visual encoder that operates in the latent domain called LatentCLIP. This greatly speed up the optimization as the latent domain is much more compacted than the pixel domain.

LatentCLIP visual encoder $\text{LatentCLIP}_{\text{vis}}$ takes a latent image and is trained to output the same feature as its pixel domain counterpart from the original CLIP visual encoder CLIP_{vis} (Fig. 4), where the model training objective is

$$1 - \cos(\text{LatentCLIP}_{\text{vis}}(L), \text{CLIP}_{\text{vis}}(I)). \tag{3}$$

$\text{LatentCLIP}_{\text{vis}}$ is initialized from a pretrained CLIP_{vis}, which the first convolution layer replaced to accommodate for the dimension of latent images.

To incorporate $\text{LatentCLIP}_{\text{vis}}$ and the CLIP text encoder $\text{CLIP}_{\text{text}}(\cdot)$ [7, 24] to the semantic loss

Algorithm 1 TiNO-Edit

```
1:  function TiNO-Edit($I, p_O, p, \mathcal{A}$)
2:      $K \in \mathbb{Z}^+, T \in [0, 1]$
3:      $t_k \leftarrow k \cdot \frac{T}{K}, k \in [0, K]$
4:      $N \sim \mathcal{N}(0, \mathbf{I})$
5:      $L \leftarrow \text{VAE}_{\text{enc}}(I)$
6:      $M = \text{MSK}(I, p_O, p, \mathcal{A})$
7:      for $w \leftarrow 1$ to $W$ do
8:          $\tilde{L}_K \leftarrow \text{FD}_K(L, N)$                        ▷ Eq. 1
9:          $\tilde{L}_K \leftarrow \tilde{L}_K \bigodot M + L \bigodot (1 - M)$
10:         for $k \leftarrow K$ to 1 do
11:             $\tilde{N}_k \leftarrow \text{UNet}_{\text{SD}}(\tilde{L}_k, t_k, p)$
12:             $\tilde{L}_{k-1} \leftarrow \text{RD}_{k,k-1}(\tilde{L}_k, \tilde{N}_k)$           ▷ Eq. 2
13:             $\tilde{L}_{k-1} \leftarrow \tilde{L}_{k-1} \bigodot M + L \bigodot (1 - M)$
14:         end for
15:         $\underset{N, t_k}{\text{argmin}}\ \mathcal{L}_{total}(L, \tilde{L}_0, p_O, p, \mathcal{A})$        ▷ Eq. 8
16:     end for
17:     return $\text{VAE}_{\text{dec}}(\tilde{L}_0)$
18: end function
```

function, we may calculate the cosine distance between the output feature and the prompt feature $\cos(\text{LatentCLIP}_{\text{vis}}(\tilde{L}_0), \text{CLIP}_{\text{text}}(p))$ [24], the distance between the original prompt and the target prompt in CLIP text domain $\text{CLIP}_{\text{text}}(p_O) - \text{CLIP}_{\text{text}}(p)$, as well as the distance between the original image and the output in CLIP image domain $(\text{LatentCLIP}_{\text{vis}}(L) - \text{LatentCLIP}_{\text{vis}}(\tilde{L}_0))$ [33] if we adopt directly from these prior methods.

However, all of them rely on the assumption that the image encoder and the text encoder are from the same CLIP pre-trained model without any modifications or fine-tuning with the same feature dimensions. This prevent different variations of SD and new concepts learned through DreamBooth [27] or Textual Inversion [13] outside what the original CLIP text encoder to be incorporated in the loss. To this end, we design a new semantic loss function that minimizes the difference between the cosine distance between original/target images in the CLIP image domain and the one between original/target prompt in the CLIP text domain as

$$\begin{aligned} &\mathcal{L}_{sem}(L, \tilde{L}_0, p_O, p) \\ =&\cos(\text{LatentCLIP}_{\text{vis}}(L), \text{LatentCLIP}_{\text{vis}}(\tilde{L}_0)) \\ &- \cos(\text{CLIP}_{\text{text}}(p_O), \text{CLIP}_{\text{text}}(p)), \end{aligned} \tag{4}$$

where $\text{CLIP}_{\text{text}}$ may include new concepts learned through DreamBooth [27] or Textual Inversion [13].

Similarly, to ensure semantic similarity to the latent representation of the reference image $L_r = \text{VAE}_{\text{enc}}(I_r)$ for reference-guided image editing, we define

$$\mathcal{L}_{ref}(L, L_r) = \cos(\text{LatentCLIP}_{\text{vis}}(L), \text{LatentCLIP}_{\text{vis}}(L_r)). \tag{5}$$

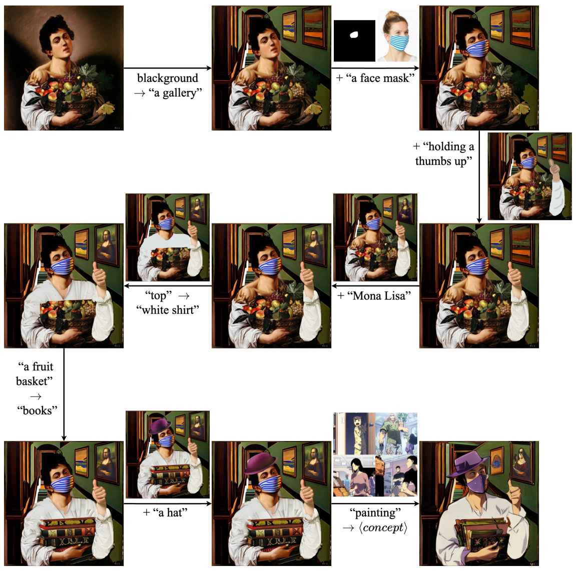


Figure 5. **Compounded image editing.** We present a compounded image-editing workflow by applying our method repeatedly on a single image. For each step, the user can perform any of the supported editing operations. Any additional information such as inputs including masks, reference images, user strokes, user-composed images, and concept images used to train custom concepts are shown next to the corresponding arrows.

Next, we want to enforce visual similarity between L and $\tilde{L}_0$. Following the same idea in LatentCLIP, we designed LatentVGG, a VGG encoder [29] that operates in the latent domain, which is initialized and trained along with a pre-trained VGG encoder. The training objective is

$$\|\text{LatentVGG}(L), \text{VGG}(I)\|_1. \tag{6}$$

After training, our perceptual loss is

$$\mathcal{L}_{perc}(L, \tilde{L}_0) = \|\text{LatentVGG}(L) - \text{LatentVGG}(\tilde{L}_0)\|_1. \tag{7}$$

To put everything together, our final loss function is

$$\begin{aligned} &\mathcal{L}_{total}(L, \tilde{L}_0, p_O, p, \mathcal{A}) \\ =&\lambda_{sem} \cdot \mathcal{L}_{sem}(L, \tilde{L}_0, p_O, p) \\ &+ \lambda_{ref} \cdot \mathcal{L}_{ref}(L, L_r) \\ &+ \lambda_{perc} \cdot \mathcal{L}_{perc}(L, \tilde{L}_0), \end{aligned} \tag{8}$$

where $\lambda_{sem} = 1$, $\lambda_{perc} = 0.5$, and $\lambda_{ref} = 1$ if applicable.

In the end, TiNO-Edit (Alg. 1) optimize N and all t_k's for a pre-defined W number of optimization steps with respect to $\mathcal{L}_{total}$, each of which consists of K timesteps as one complete denoising process.

5. Experiments

TiNO-Edit uses SD v2.1 [25] with $K = 10$ denoising steps, starting timestep value $T = 0.75$, and the number of op-

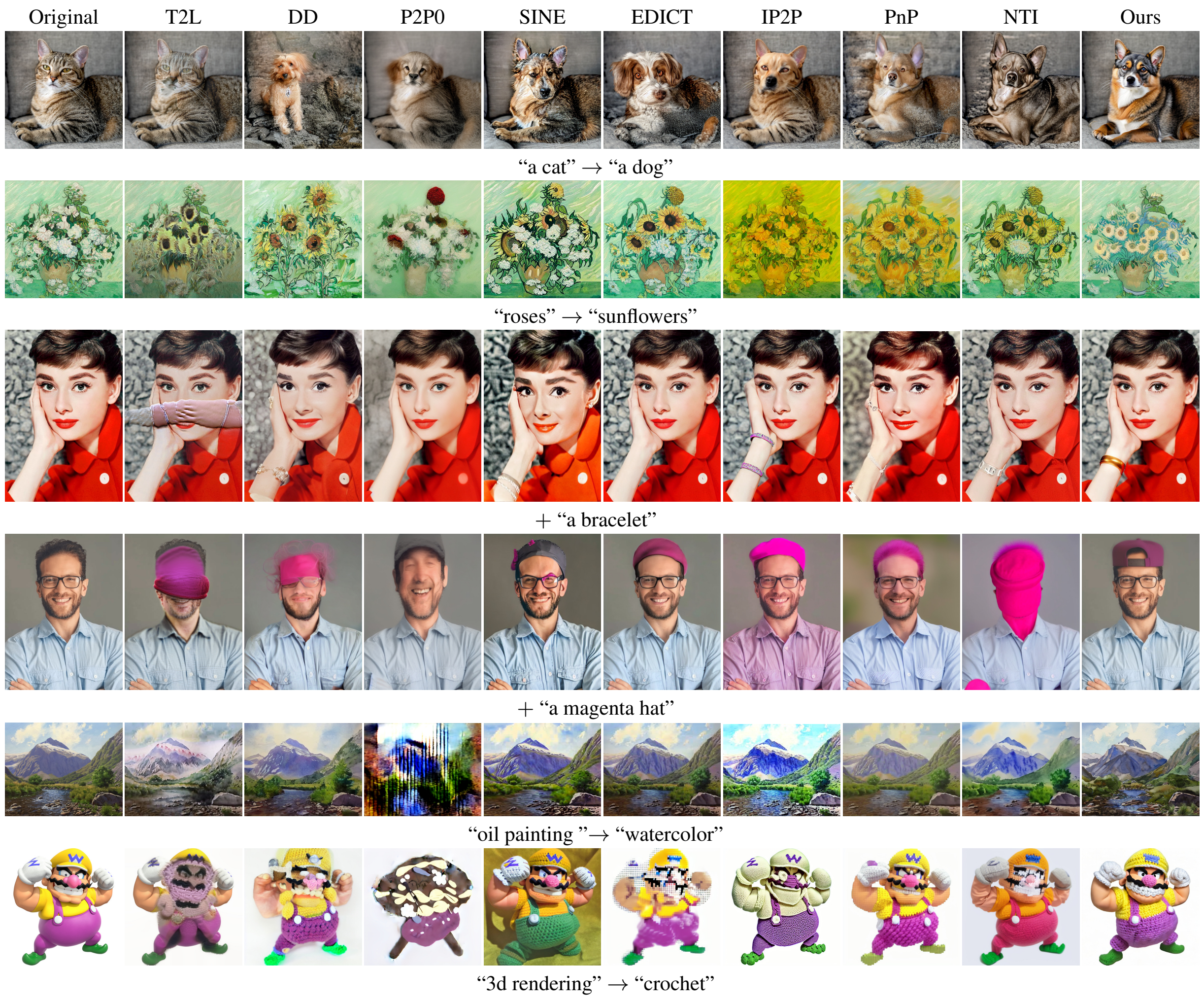


Figure 6. **Pure text-guided image editing comparison.** We compare with the following baselines: Text2LIVE (T2L) [2], DisentanglementDiffusion (DD) [33], Pix2Pix-Zero (P2P0) [22], SINE [36], EDICT [32], InstructPix2Pix (IP2P) [3], Plug-and-Play (PnP) [31], and Null-text Inversion (NTI) [21]. The user indicates the desired editing via text, where the part that reflects the edit is shown below each row.

timization steps $W = 50$. To use LatentCLIP and LatentVGG for our loss function, each model has been trained on a subset of LAION-5B [28] datasets for 10^5 iterations with a batch size of 16. We use the AdamW [16] optimizer with a learning rate of $lr = 1\text{e-}5$.

We use two AdamW [16] optimizers, one for t_k's with $lr = 1$ and the other for N with $lr = 0.005$. Our method can also run on DreamBooth (DB) [27] or Textual Inversion (TI) [13], where we use the same SD and noise scheduler DB/TI is trained with to avoid performance degradation.

5.1. Compounded Image Editing

We present a compounded image editing workflow by applying our method repeatedly on a single image with different operations(Fig. 5). For each step, the user can perform any of the supported editing operations, which gives the user creative control over how they want to edit images.

5.2. Qualitative comparisons

Pure text-guided image editing. We compare TiNO-Edit with other text-guided image editing baselines, including Text2LIVE (T2L) [2], DisentanglementDiffusion (DD) [33], Pix2Pix-Zero (P2P0) [22], SINE [36], EDICT [32], InstructPix2Pix (IP2P) [3], Plug-and-Play (PnP) [31], and Null-text Inversion (NTI) [21] and test various use cases (Fig. 6). While some baselines generate good results in sevaeral scenarios (e.g., EDICT rows 2, 4; NTI rows 2, 3, 5), many still pose various issues, including incohesive edit (T2L rows 3, 4, 5; DD row 4), failing to add new objects (P2P0, EDICT, PnP for row 3), concept leaking (IP2P row 2: image becomes yellow; row 3: shirt is magenta-colored; NTI row 4: man gets a hat texture), and altering other attributes (P2P0, SINE, PnP row 4; NTI row 6). On the other hand, our method is robust across use cases.

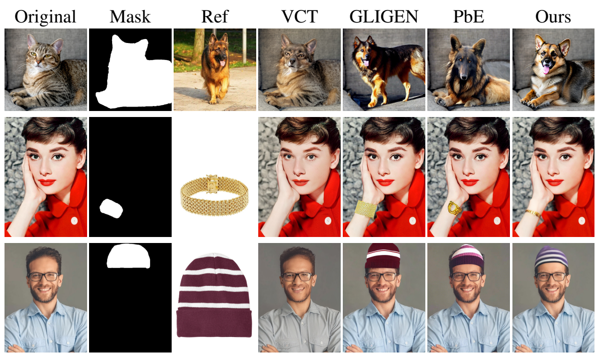


Figure 7. **Comparison with reference-guided image editing baselines.** We compare our method with Visual Concept Translator (VCT) [6], GLIGEN [17], and Paint-by-Example (PbE) [34], where the masks (Mask) indicate edit regions where the object in the reference image (Ref) should be included.

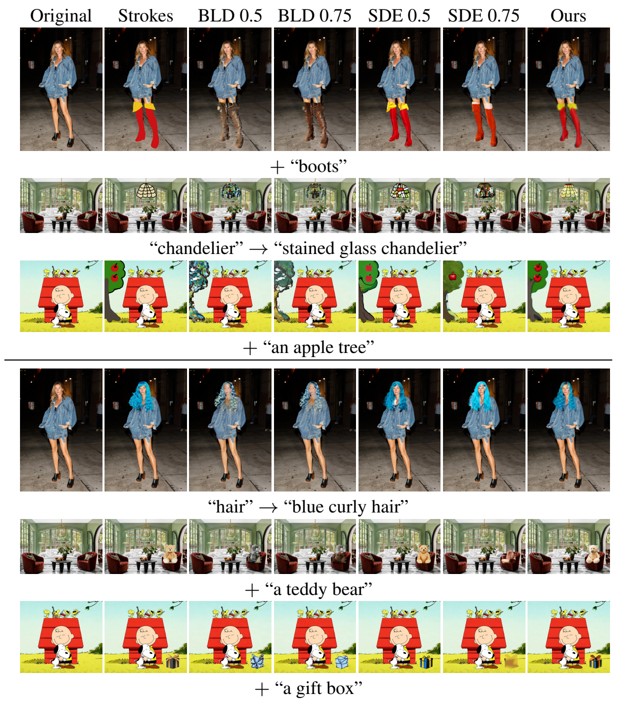


Figure 8. **Stroke-based image editing and image composition baseline comparison.** We compare to Blended Latent Diffusion (BLD) [1] and SDEdit (SDE) [20] for stroke-guided image editing (row 1-3) and image composition (row 4-6). BLD and SDEdit both accept a starting timestep T as input, and we test two values 0.5 and 0.75 (shown next to the method names).

Reference-guided image editing. For reference-guided image editing, we compare our method with Visual Concept Translator (VCT) [6], GLIGEN [17], and Paint-by-Example (PbE) [34] (Fig. 7). our method is better at preserving details (a dog with its tongue sticking out, white stripes on the hat) when maintaining other image regions. In contrast, VCT failed to add objects, GLIGEN does not compose edited objects to the original images (dog standing on the sofa) and PbE generates objects that are similar to the references at a high level without keeping their details.

	T2L	DD	P2P0	SINE	EDICT	IP2P	PnP	NTI	Ours
CLIP-T	0.299	0.316	0.271	0.314	0.327	0.315	0.321	0.316	**0.328**
CLIP-I	0.879	0.853	0.793	0.912	0.898	0.845	0.835	0.843	**0.924**
DINO-I	0.864	0.862	0.776	0.805	0.838	0.828	0.756	0.799	**0.874**

Table 1. Pure text-guided image editing qualitative evaluation.

	VCT	GLIGEN	PbE	Ours
CLIP-T	0.267	0.295	0.300	**0.311**
CLIP-I	0.899	0.814	0.841	**0.928**
CLIP-I$_r$	0.537	0.616	0.570	**0.674**
DINO-I	**0.919**	0.770	0.813	0.869

Table 2. Reference-guided image editing qualitative evaluation

	BLD 0.5	BLD 0.75	SDEdit 0.5	SDEdit 0.75	Ours
CLIP-T	**0.298**	0.287	0.295	0.295	**0.298**
CLIP-I	0.930	0.924	0.897	0.953	**0.959**
CLIP-I$_s$	0.956	0.927	0.986	0.956	**0.987**
DINO-I	0.953	0.959	0.961	0.973	**0.978**

Table 3. Stroke-guided image editing qualitative evaluation

	BLD 0.5	BLD 0.75	SDEdit 0.5	SDEdit 0.75	Ours
CLIP-T	0.273	0.273	0.294	0.267	**0.298**
CLIP-I	0.955	0.950	0.903	0.937	**0.958**
CLIP-I$_c$	0.943	0.945	0.965	0.964	**0.978**
DINO-I	0.977	0.975	0.972	0.973	**0.983**

Table 4. Image composition qualitative evaluation

Stroke-guided image editing. We compare to Blended Latent Diffusion (BLD) [1] and SDEdit (SDE) [20] for stroke-guided image editing (row 1-3 of Fig. 8), both of which take a starting timestep as input. BLD largely generates results that do not align with the strokes and the same problem appears in SDEdit. On the other hand, TiNO-Edit can generate objects from strokes that blend well with the original.

Image composition. We also compare with BLD and SDEdit for image composition use cases (Fig. 8 row 4-6), where TiNO-Edit refine user-composed images and generate cohesive and nature-looking outputs (Fig. 8) such as making the teddy bear sit on the sofa rather than floating in front of it. Again, BLD changes the user input drastically. SDEdit with the starting timestep of 0.75 largely omits the new object that should be added to the results. While SDEdit with the starting timestep of 0.5 generates better results, the output may not look cohesive.

5.3. Quantitative Comparisons

We created a test set of 200 samples in the form of $(I, p_O, p, I_r, I_s, I_c)$, where we ran each method for each sample and report the following popular metrics averaged across the test set (Tabs. 1 to 4):

- CLIP-T: $\cos(\text{CLIPvis}(\tilde{I}), \text{CLIPtext}(p))$
- CLIP-I: $\cos(\text{CLIP}_{\text{vis}}(\tilde{I}), \text{CLIP}_{\text{vis}}(I))$
- CLIP-I$_*$: $\cos(\text{CLIP}_{\text{vis}}(\tilde{I}), \text{CLIP}_{\text{vis}}(I_*))$, $I_* \in \{I_r, I_s, I_c\}$
- DINO-I: $\cos(\text{DINO}_{\text{vis}}(\tilde{I}), \text{DINO}_{\text{vis}}(I))$

where $\tilde{I}$ is the output, DINO_{vis} refers to DINO ViT [5] which can be used to evaluate image perceptual alignment. Our method outperforms other methods in most metrics, including DINO-I, a non-CLIP-based metric not used during training. In Tab. 2, VCT scores the highest in DINO-I as its outputs are very similar to the original images without respecting the target prompts/images (Fig. 7), which is also supported by the fact that it gets the worst CLIP-T score.

5.4. Ablation studies

Change in timesteps and noise throughout optimization. We look at how t_k's and N change across W optimization steps and average the values across our test set in Fig. 9. Timesteps earlier in the diffusion process are changed more compared to timesteps later in the diffusion process because the former has a larger effect on the editing result. The range and the mean of N is quite stable over time because they are optimized with respect to a lower learning rate and handle more local changes with respect to the changing t_k's.

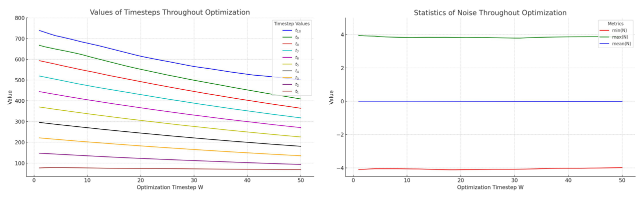


Figure 9. **Changes in timesteps and noise through optimization.** We plot the values of t_k's alongside the minimum, maximum, and mean of N across W optimization steps. This indicates that N remains within the noise distribution that SD has been learned, preventing potential degradation of the output image.

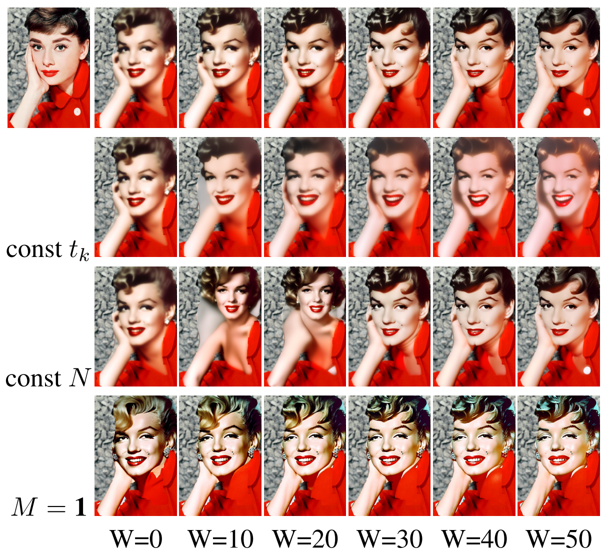


Figure 10. **Effect of optimizing timesteps, noise, and using the masking mechanism.** Optimizing both t_k's and N leads to optimize results across different number of optimization steps.

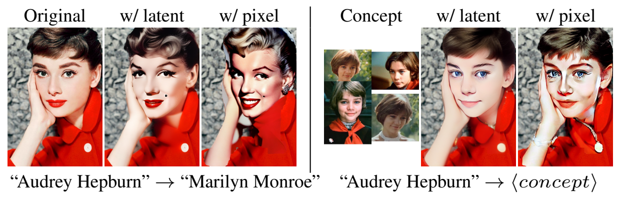

Original
w/ latent
w/ pixel
Concept
w/ latent
w/ pixel
"Audrey Hepburn" → "Marilyn Monroe"
"Audrey Hepburn" → $\langle concept \rangle$

Figure 11. **Effect of LatentCLIP and LatentVGG on image editing quality and custom concepts.** Compared with the original CLIP [24] and VGG [29] that operate in pixel spaces ("w/ pixel"), using our LatentCLIP and LatentVGG ("w/ latent") is not only more efficient but leads to better quality outputs.

Stability in N also means that we don't need to worry about N going outside the noise distribution SD has learnt and cause performance degrade.

Effect of optimizing timesteps, noise, and using the masking mechanism. We first study the effect of optimizing timesteps, noise, and using the masking mechanism by generating results with different optimization configuration with a fixed seed (Fig. 10). As we can see, TiNO-Edit is the most robust across various W. Keeping t_k's constant causes the outputs to degrade over time because when the timestep values are sub-optimal, N has to change more drastically to over-compensate for our loss function, which pushes it away from the Gaussian noise distribution that SD learns. Omitting optimizing N has a smaller effect on the results with larger W but is more prone to artifacts (the shadow around the coat collar). Moreover, when W is small, we may get drastically different images compared to the original. Lastly, when the editing region is not constrained by the mask M, the global structure of the image may get changed entirely. This suggest that optimizing both t_k's and N is necessary for optimal results.

Effect of LatentCLIP and LatentVGG. We study the effect of using LatentCLIP and LatentVGG compared to using the original CLIP [24] and VGG [29] on a NVIDIA A6000. In terms of image editing quality, using our LatentCLIP and LatentVGG leads to high-quality image editing results with better alignment with the original image and less artifacts (Fig. 11 row 1) while performing better with custom concepts in Stable Diffusion variants such as Textual Inversion [13] (Fig. 11 row 2).

6. Conclusion

In this paper, we have presented TiNO-Edit, an image editing method with pre-trained Stable Diffusion models by optimizing diffusion timesteps and input noise. We have designed a set of loss functions that greatly reduce the computation resources required for optimization. Comparing TiNO-Edit with various task-specifically baselines, our method is able to generalize across specific image editing tasks and produce superior results.

Appendix

A. Additional Results

Here we show more results, including pure text-guided image editing (Fig. 12), reference-guided image editing (Fig. 13), stroke-guided image editing (Fig. 14), and image composition (Fig. 15). Our method can be applied for any of the above use cases with custom concepts in DreamBooth [27] and Textual Inversion [13] (Fig. 16).

B. Additional Ablations

B.1. Effect of user inputs

We explore the effect of user inputs on the editing results by looking at a stroke-guided image editing use case (Fig. 17). Here the user provides input strokes and intends to convert them into “a doll”. When the input strokes contain a smiley face, our method successfully converts it into a realistic-looking face of a doll. When the smiley face strokes are omitted in the input strokes, our method interprets this as a side view of the doll’s face, generating results accordingly.

B.2. Effect of noise schedulers

When editing with custom concepts in DreamBooth (DB) [27] or Textual Inversion (TI) [13], we use the same noise scheduler as when these concepts are trained with their respective models rather than using the default DDIM scheduler [30] because the concepts can get degraded or lost with a different noise scheduler as shown in Fig. 18.

B.3. Effect of LatentCLIP and LatentVGG

We study the computational requirement of our method using LatentCLIP and LatentVGG compared to using the original CLIP [24] and VGG [29] by averaging the GPU memory usage and optimization time of our method across 50 trials (Tab. 5). Our method with its LatentCLIP and LatentVGG requires only around 50% of GPU memory and time compared to using the original CLIP and VGG, as the latter operates in the image pixel domain which is much larger than the SD latent domain.

	w/ latent	w/ pixel
Average GPU RAM usage (GB)	22	41
Average optimization time (s)	63	150

Table 5. **Effect of LatentCLIP and LatentVGG on computational requirement**. We study the computational requirement of our method using LatentCLIP and LatentVGG (col “w/ latent”) as well as the original CLIP [24] and VGG [29] (col “w/ pixel”) on NVIDIA A6000 (48GB), where we average the GPU memory usage and optimization time across 50 trials. Our method with its LatentCLIP and LatentVGG requires only around 50% of GPU memory and time compared to using the original CLIP and VGG.

References

[1] Omri Avrahami, Ohad Fried, and Dani Lischinski. Blended latent diffusion. *ACM Transactions on Graphics (TOG)*, 42 (4):1–11, 2023. 2, 7

[2] Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. Text2LIVE: Text-driven layered image and video editing. In *European Conference on Computer Vision (ECCV)*, pages 707–723. Springer, 2022. 6

[3] Tim Brooks, Aleksander Holynski, and Alexei A Efros. InstructPix2Pix: Learning to follow image editing instructions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18392–18402, 2023. 2, 6

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:1877–1901, 2020. 2

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *International Conference on Computer Vision (ICCV)*, 2021. 8

[6] Bin Cheng, Zuhao Liu, Yunbo Peng, and Yue Lin. General image-to-image translation with one-shot image guidance. In *International Conference on Computer Vision (ICCV)*, pages 22736–22746, 2023. 2, 3, 7

[7] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2829, 2023. 4

[8] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. DiffEdit: Diffusion-based semantic image editing with mask guidance. *arXiv preprint arXiv:2210.11427*, 2022. 2

[9] Katherine Crowson, Stella Biderman, Daniel Kornis, Dashiell Stander, Eric Hallahan, Louis Castricato, and Edward Raff. VQGAN-CLIP: Open domain image generation and editing with natural language guidance. In *European Conference on Computer Vision (ECCV)*, pages 88–105. Springer, 2022. 4

[10] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:8780–8794, 2021. 2

[11] Dave Epstein, Allan Jabri, Ben Poole, Alexei A. Efros, and Aleksander Holynski. Diffusion self-guidance for controllable image generation. *arXiv preprint arXiv:2306.00986*, 2023. 3

[12] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883, 2021. 3, 4, 14

[13] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or.

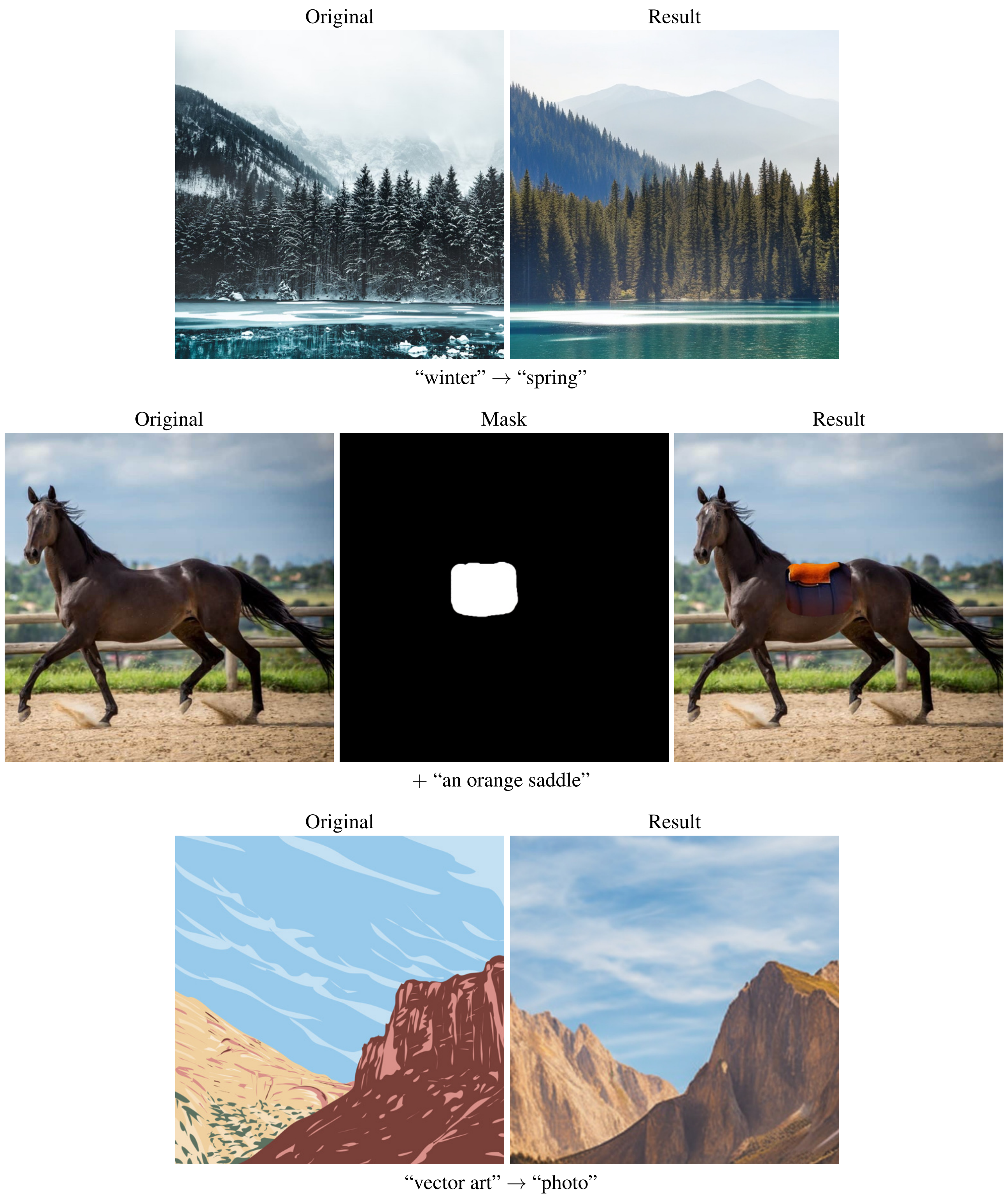


Figure 12. **Pure text-guided image editing examples.** We can use text to perform various image editing operations including changing object attributes (row 1), adding objects (with a mask indicating the location; row 2), or changing image style (row 3). The desired editing is indicated via text prompts, where the part that reflects the editing is shown below each sample.

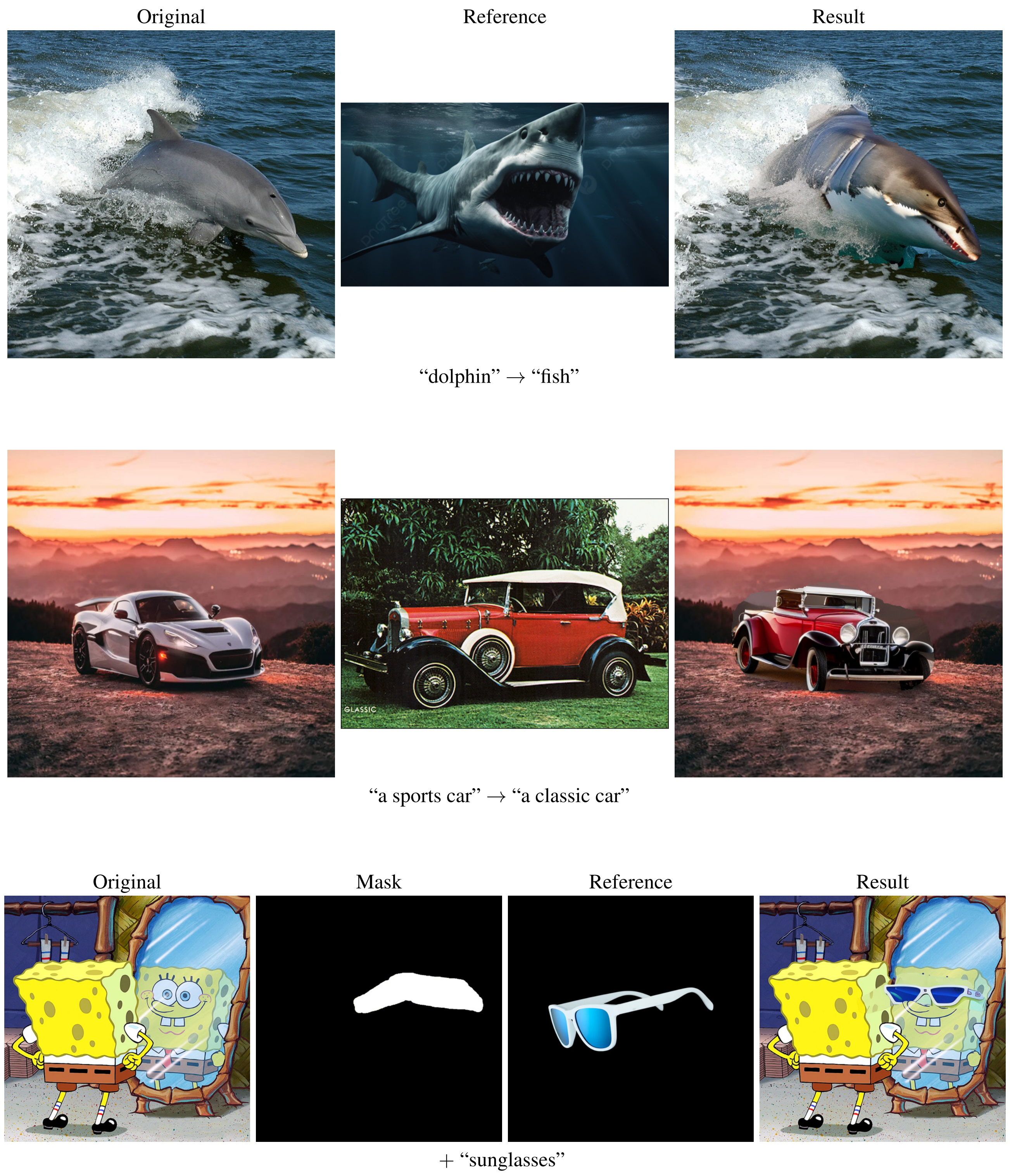


Figure 13. **Reference-guided image editing examples.** Our method can take a reference image and either replace objects (row 1,2) or add objects in the image region indicated by the masks (row 3) to the corresponding objects in the reference image. The desired editing is indicated via text prompts, where the part that reflects the editing is shown below each sample.

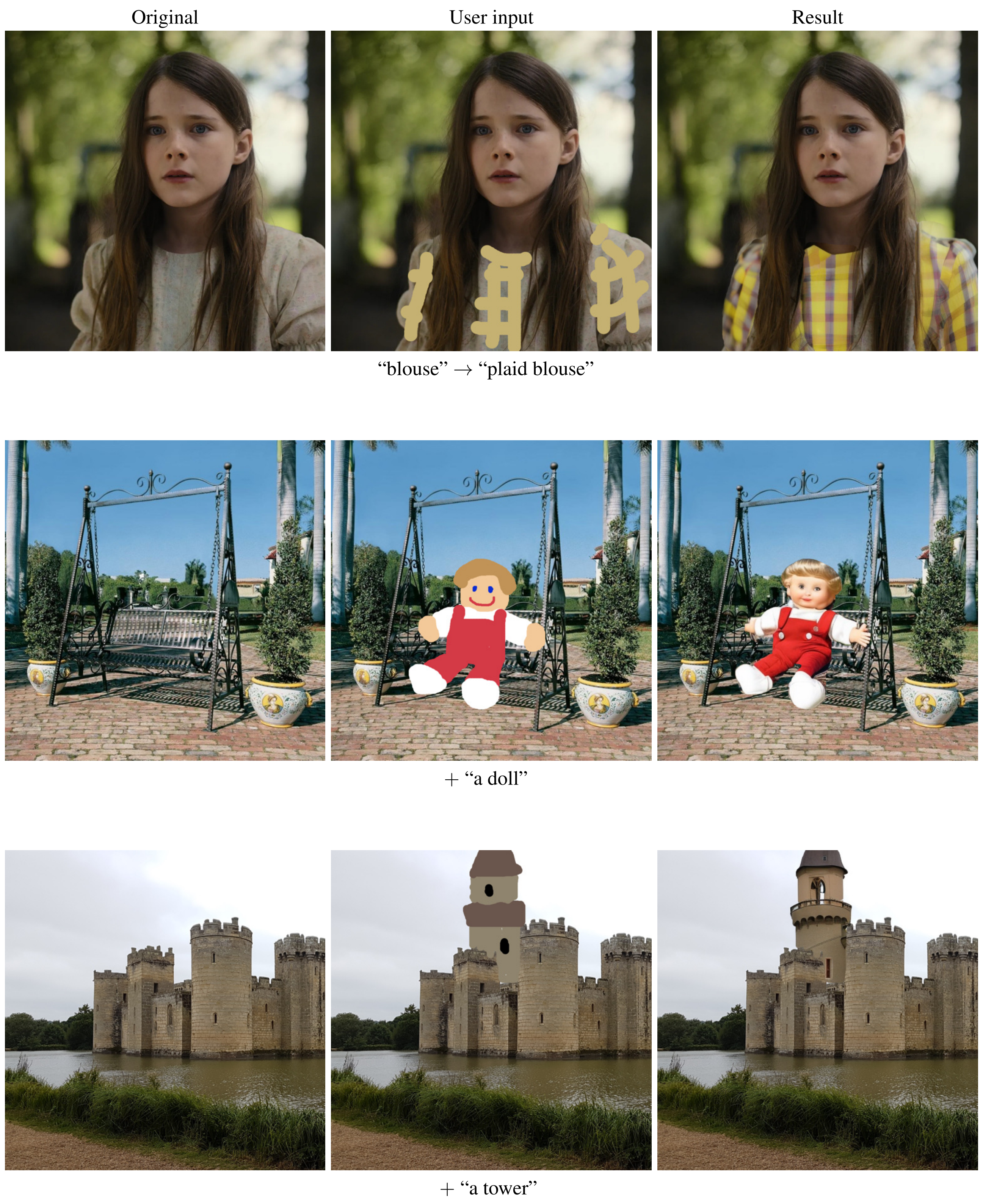


Figure 14. **Stroke-guided image editing examples.** Our method is able to take user strokes and edit the image accordingly, where the strokes will be converted based on the input text prompts, where the part that reflects the editing is shown below each sample. It can change existing objects in the image (row 1) as well as add new objects (rows 2-3) to the images.

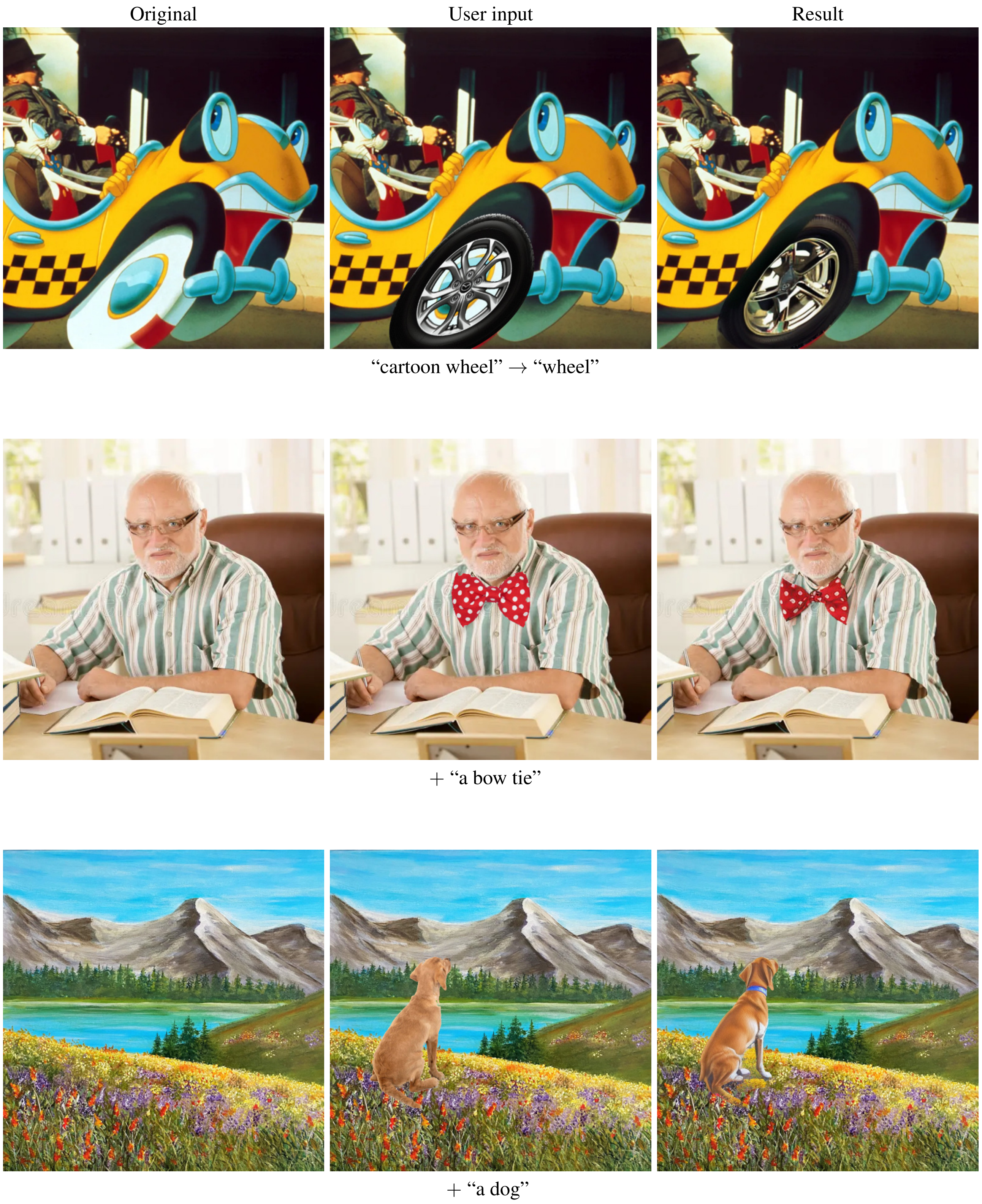


Figure 15. **Image composition examples.** Our method is able to take a user-composed image and harmonize it. For example, we can adjust the position and the lighting of the composed car wheel (row 1) and the bow tie (row 2) automatically so they blend naturally as part of the image rather than floating on top of their respective images. Similarly, we can add a photo of a dog to a drawing and change its style to fit more closely to the drawing itself.

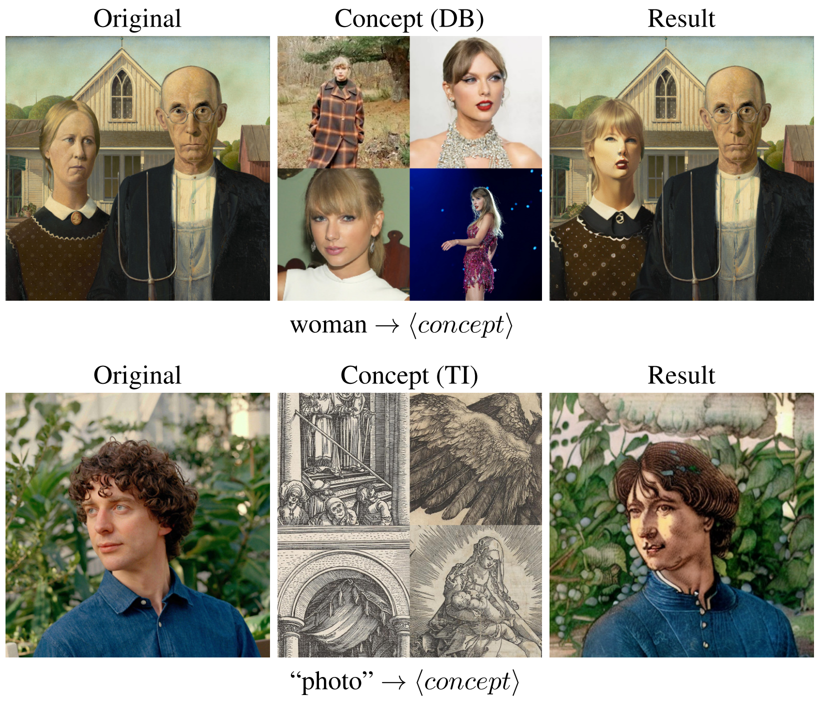

Original
Concept (DB)
Result
woman $\rightarrow \langle concept \rangle$
Original
Concept (TI)
Result
"photo" $\rightarrow \langle concept \rangle$

Figure 16. **Image editing with DreamBooth and Textual Inversion.** Our method can be applied with DreamBooth (DB) [27] or Textual Inversion (TI) [13] and incorporate their corresponding custom concepts in the image editing capabilities. The desired editing is indicated via text prompts with the token corresponding to the concept, where the part that reflects the editing is shown below each sample and the token is denoted as $\rightarrow \langle concept \rangle$.

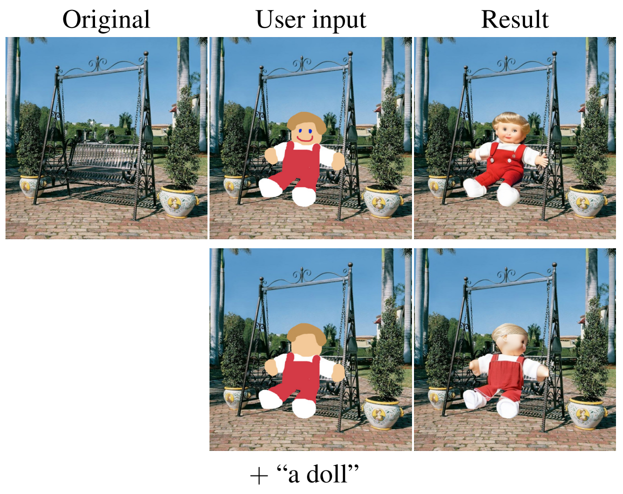


Figure 17. **Effect of user inputs.** We can see the effect of user inputs on editing results in a stroke-guided image editing use case. Here the user provides input strokes and intends to convert them into "a doll". When the strokes contain a smiley face, it is converted into a realistic-looking face of a doll. When the smiley face strokes are omitted, our method interprets it as the side view of the face of the doll.

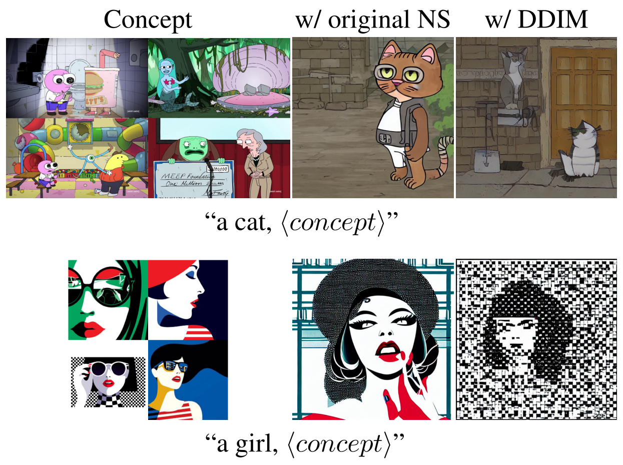

Concept
w/ original NS
w/ DDIM
"a cat, $\langle concept \rangle$"
"a girl, $\langle concept \rangle$"

Figure 18. **Effect of noise schedulers with custom concepts.** When editing with custom concepts in DreamBooth (DB) [27] and/or Textual Inversion (TI) [13], we use the same noise scheduler they are trained with because the concepts can get lost otherwise. Take a look at Stable Diffusion [12] T2I outputs (input text prompts shown below each sample) using DB (row 1) or TI (row 2) with their original noise schedulers (NS) vs. the DDIM [30] scheduler. The style of outputs match more closely to the concepts using the original noise scheduler.

An Image is Worth One Word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022. 1, 2, 5, 6, 8, 9, 14

[14] Shanghua Gao, Zhijie Lin, Xingyu Xie, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. EditAnything: Empowering unparalleled flexibility in image editing and generation. In *ACM International Conference on Multimedia (ACMMM)*, 2023. 2

[15] HuggingFace. HuggingFace Stable diffusion image-to-image, 2023. https://huggingface.co/docs/diffusers/api/pipelines/stable_diffusion/img2img. 3

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[17] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. GLIGEN: Open-set grounded text-to-image generation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22511–22521, 2023. 2, 7

[18] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7086–7096, 2022. 4

[19] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023. 2

[20] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2022. 2, 4, 7

[21] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6038–6047, 2023. 2, 3, 6

[22] Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. In *SIGGRAPH*, pages 1–11, 2023. 2, 3, 6

[23] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 2

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021. 2, 4, 5, 8, 9

[25] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 1, 2, 3, 5

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 2, 3

[27] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. DreamBooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22500–22510, 2023. 1, 2, 5, 6, 9, 14

[28] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. LAION-5B: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:25278–25294, 2022. 6

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5, 8, 9

[30] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2, 3, 9, 14

[31] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1921–1930, 2023. 2, 3, 6

[32] Bram Wallace, Akash Gokul, and Nikhil Naik. EDICT: Exact diffusion inversion via coupled transformations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22532–22541, 2023. 2, 6

[33] Qiucheng Wu, Yujian Liu, Handong Zhao, Ajinkya Kale, Trung Bui, Tong Yu, Zhe Lin, Yang Zhang, and Shiyu Chang. Uncovering the disentanglement capability in text-to-image diffusion models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1900–1910, 2023. 2, 3, 4, 5, 6

[34] Binxin Yang, Shuyang Gu, Bo Zhang, Ting Zhang, Xuejin Chen, Xiaoyan Sun, Dong Chen, and Fang Wen. Paint by Example: Exemplar-based image editing with diffusion models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18381–18391, 2023. 2, 7

[35] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *International Conference on Computer Vision (ICCV)*, pages 3836–3847, 2023. 2

[36] Zhixing Zhang, Ligong Han, Arnab Ghosh, Dimitris N Metaxas, and Jian Ren. SINE: Single image editing with text-to-image diffusion models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6027–6037, 2023. 2, 3, 6