Supplementary Material for "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

In this supplementary document, we present additional results of various image editing capabilities that TiNO-Edit supports, as detailed in Section A. Then we discuss the influence of user inputs on the outcomes of image editing (Sec. B.1) and study the effects of varying noise schedulers on the quality of the results when integrating custom concepts in DreamBooth [4] and/or Textual Inversion [2] (Sec. B.2). Lastly, we offer a comparative analysis of the computational demands of employing our method with LatentCLIP and LatentVGG versus the traditional CLIP and VGG models in Section B.3.

A. Additional Results

We show more results of various image editing capabilities that TiNO-Edit supports, including pure text-guided image editing (Fig. 1), reference-guided image editing (Fig. 2), stroke-guided image editing (Fig. 3), and image composition (Fig. 4). Our method can be applied for any of the above use cases with custom concepts in DreamBooth [4] and Textual Inversion [2] (Fig. 5). As we can see, our method can be applied for any of the above use cases and is capable of generating realistic results.

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. 6). 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.

*Corresponding author email: xchen774@ucsb.edu

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

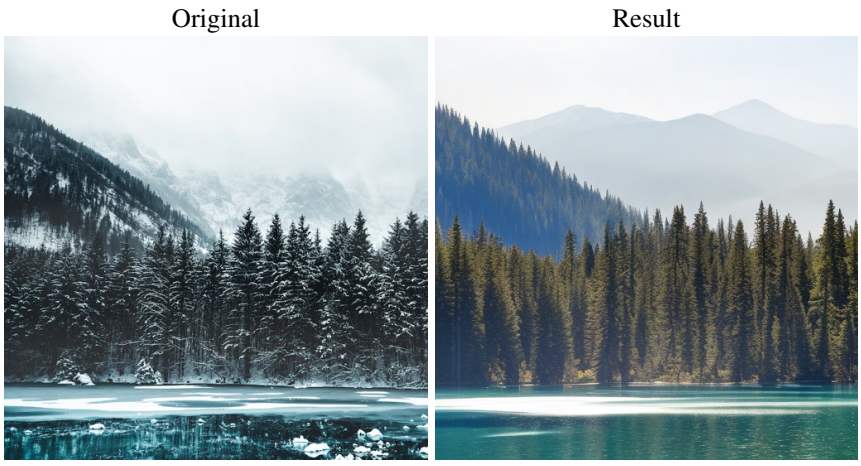
Table 1. **Effect of LatentCLIP and LatentVGG on computational resource requirement**. We study the computational resource requirement of our method using our LatentCLIP and LatentVGG (col "w/ latent") as well as the original CLIP [3] and VGG [5] (col "w/ pixel") on NVIDIA A6000 (48GB), where we average 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.

B.2. Effect of noise schedulers

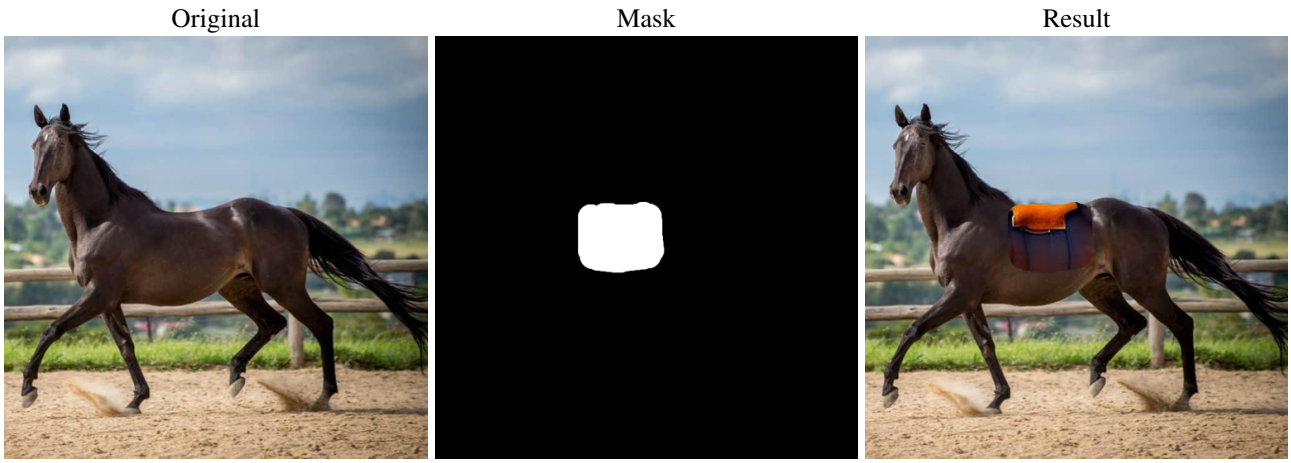
When editing with custom concepts in DreamBooth (DB) [4] or Textual Inversion (TI) [2], we use the same noise scheduler as when these concepts are trained with their respective models rather than using the default DDIM scheduler [6] because the concepts can get degraded or lost with a different noise scheduler as shown in Fig. 7.

B.3. Effect of LatentCLIP and LatentVGG

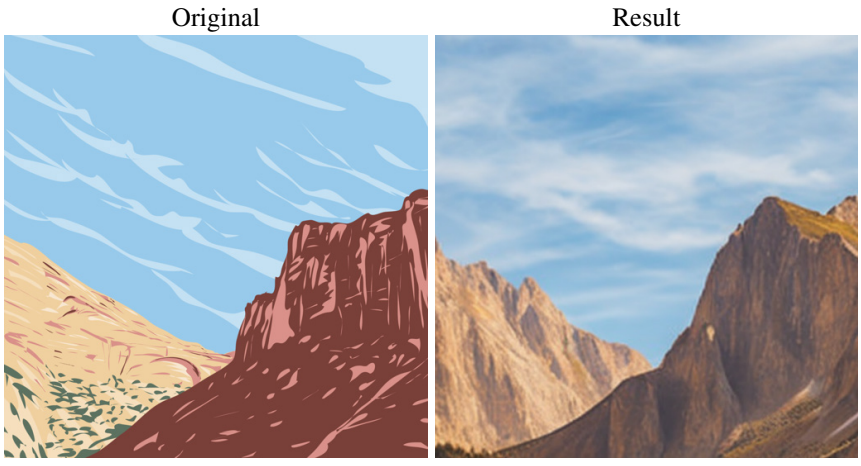
We study the computational resource requirement of our method using our LatentCLIP and LatentVGG compared to using the original CLIP [3] and VGG [5] by averaging the GPU memory usage and optimization time of our method across 50 trials (Tab. 1). Our method with its LatentCLIP and LatentVGG requires only around 50% of GPU memory and time (22 GB and 63 seconds) compared to using the original CLIP and VGG (41 GB and 150 seconds), as the latter operates in the image pixel domain which is much larger than the SD latent domain.



"winter" → "spring"



+ "an orange saddle"



"vector art" → "photo"

Figure 1. **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.

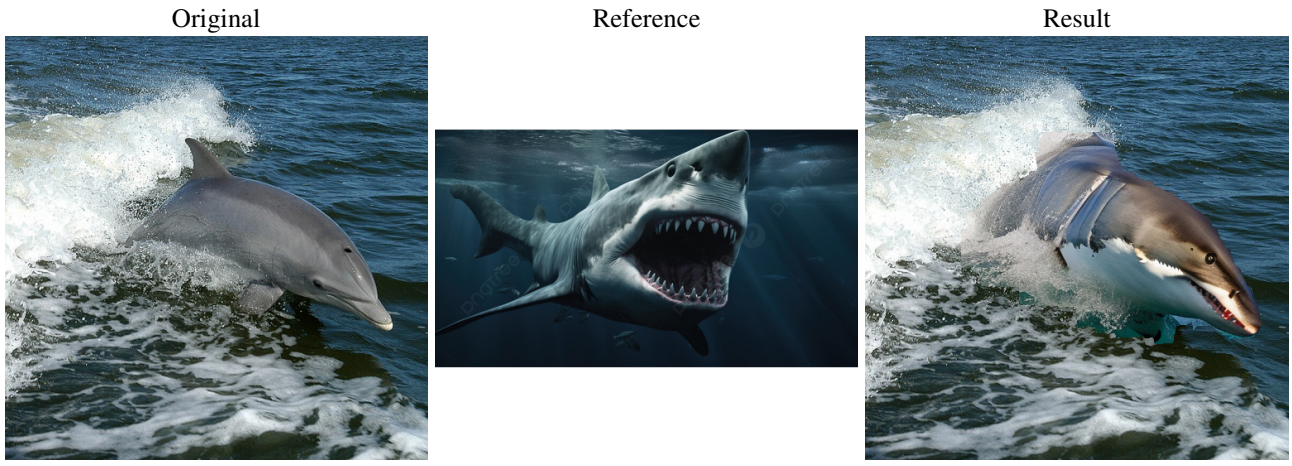


Figure 2. **Reference-guided image editing examples.** Our method can take a reference image and either replace objects (row 1,2) or add objects in the image region indicated by the masks (row 3) to the corresponding objects in the reference image. The desired editing is indicated via text prompts, where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

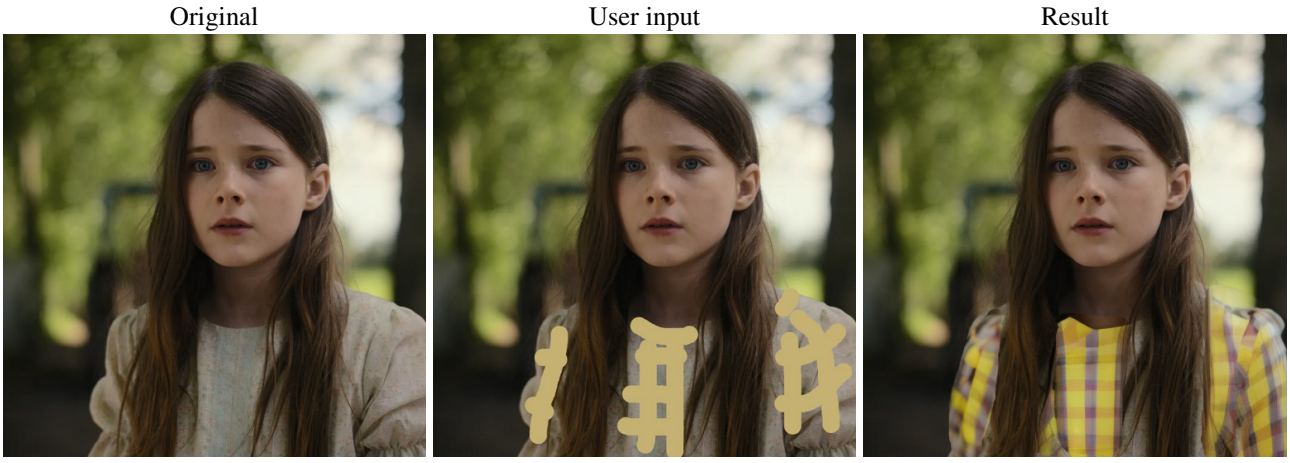


Figure 3. **Stroke-guided image editing examples.** Our method is able to take user strokes and edit the image accordingly, where the strokes will be converted based on the input text prompts, where the part that reflects the editing is shown below each sample. It can change existing objects in the image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

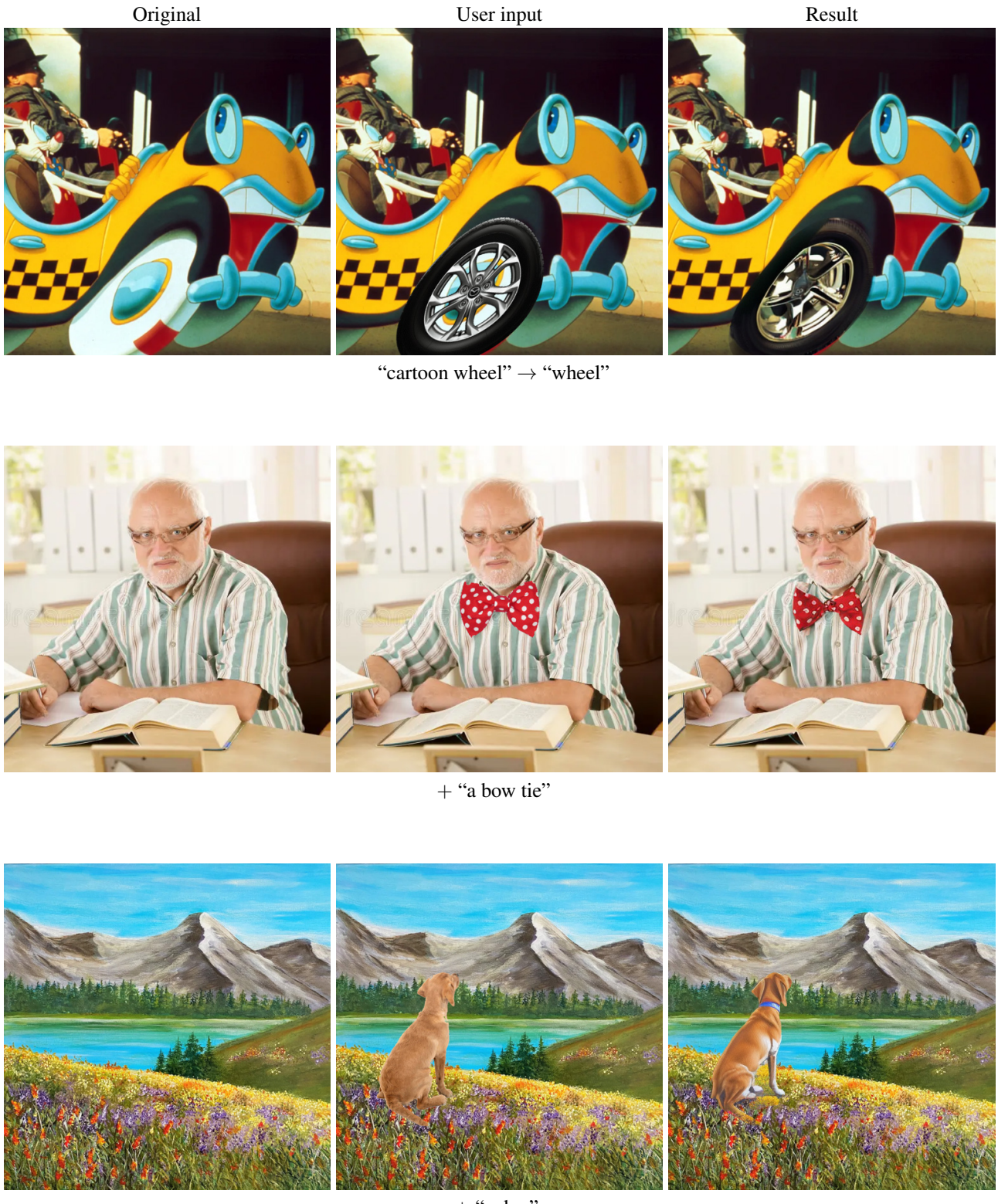


Figure 4. **Image composition examples.** Our method is able to take a user-composed image and harmonize it. For example, we can adjust the position and the lighting of the composed car wheel (row 1) and the bow tie (row 2) automatically so they blend naturally as part of the image rather than floating 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.

Figure 5. **Image editing with DreamBooth and Textual Inversion.** Our method can be applied with DreamBooth (DB) [4] or Textual Inversion (TI) [2] 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 → ⟨*concept*⟩.

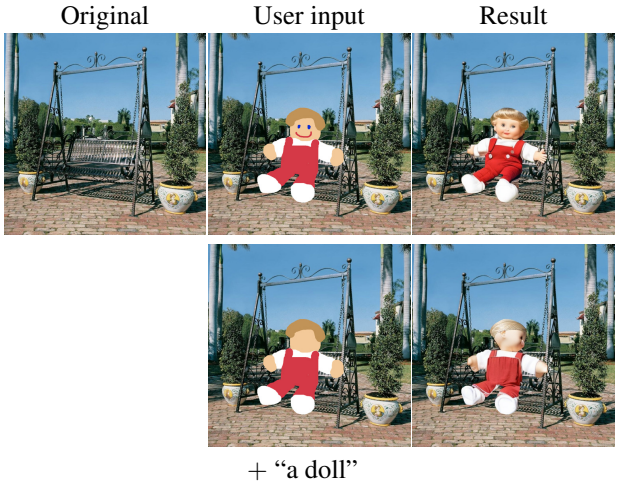


Figure 6. **Effect of user inputs.** We can see the effect of user inputs on editing results in a stroke-guided image editing use case. Here the user provides input strokes and intends to convert them into “a doll”. When the strokes contain a smiley face, it is converted into a realistic-looking face of a doll. When the smiley face strokes are omitted, our method interprets it as the side view of the face of the doll.

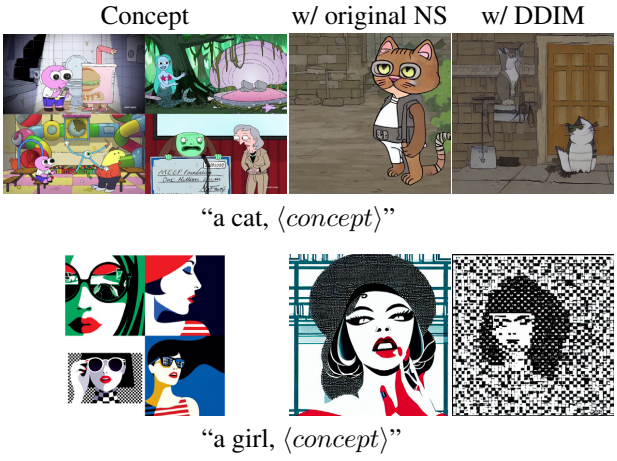


Figure 7. **Effect of noise schedulers with custom concepts.** When editing with custom concepts in DreamBooth (DB) [4] and/or Textual Inversion (TI) [2], we use the same noise scheduler they are trained with because the concepts can get lost otherwise. Take a look at Stable Diffusion [1] T2I outputs (input text prompts shown below each sample) using DB (row 1) or TI (row 2) with their original noise schedulers (NS) vs. the DDIM [6] scheduler. The style of outputs match more closely to the ones of images the concepts are trained on using the original noise scheduler.

References

[1] 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. 6

[2] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An Image is Worth One Word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022. 1, 6

[3] 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. 1

[4] 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, 6

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[6] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 1, 6