Visual Instruction Inversion: Image Editing via Visual Prompting
— *Supplementary Material* —

Anonymous Author(s)
Affiliation
Address
email

This document provides additional information complementing the main paper. First, we compare to Textual Inversion in Sec. A. Then, in Sec. B, we provide additional qualitative comparisons to Imagic [3] and Null-text Inversion [5]. Finally, in Sec. C, we provide the implementation details of our method, along with results obtained using a variant of our approach - instruction concatenation. This variant allows users to add extra information into the learned instruction.

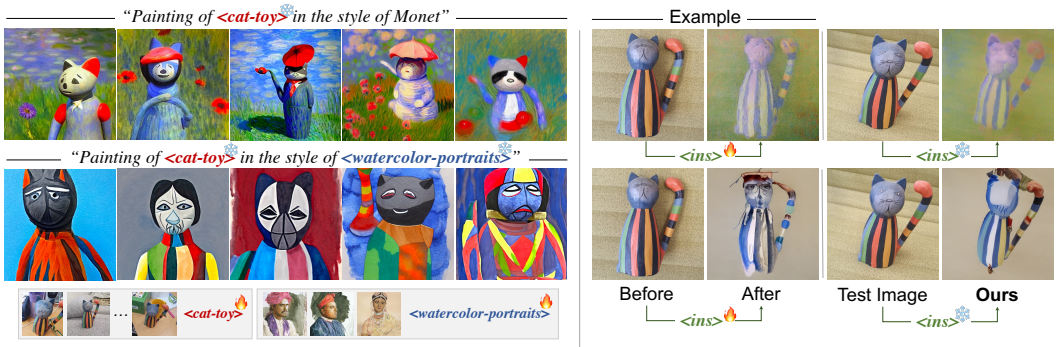

Figure 1: **Ours vs. Textual Inversion**. (a) Textual Inversion inverts a visual concept or object (e.g., a particular `<cat-toy>`) into a word embedding. This optimized word embedding can then be combined with a textual description to generate novel scenes. (b) Our Visual Instruction Inversion learns the transformation `<ins>`: "before" → "after" in a given before-and-after image pair. This learned instruction can then be applied to new test images to perform the same edit.

A Textual Inversion vs. Visual Instruction Inversion

Textual Inversion [2, 6] is a method to invert a visual concept into a corresponding representation in the language space. In particular, given (i) a text-to-image pre-trained model and (ii) some images describing a visual concept (e.g., a particular kind of toy; Figure 1 bottom row), Textual Inversion learns new "words" in the embedding space of the text-to-image model to represent those visual concepts. Once these "words" are learned for that concept, they can be plugged into arbitrary textual descriptions, just like other English words, which can then be used to create the target visual concept in different contexts. Instead of learning the representation for an isolated visual concept, our approach (Visual Instruction Inversion), learns the *transformation* from a before-and-after image pair. This learned transformation is then applied to a test image to achieve similar edit "before" → "after".

Applicability of Textual Inversion for image editing. Given these differences with our proposed method, we now try to see if Textual Inversion can be used for image editing. Textual Inversion can

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

generate a "painting of a <cat-toy> in the style of Monet" by using the learned word <cat-toy> from example images (Figure 1a, Row 1). However, the synthesized images often only capture the essence of the objects, and disregard the details of the input images. As a result, textual inversion is suitable for novel scene composition, but is not effective for image editing.

On the other hand, our Visual Instruction Inversion does not learn novel token representations for objects or concepts. Instead, we learn the edit instruction from before-and-after pairs, which can be applied to any test image to obtain corresponding edits. This allows us to achieve fine-grained control over the resulting images. For example, by providing a photo of <cat-toy>, one before and one in a specific impressionist style, we learn the transformation from before to impressionist, denoted as <ins>. Once learned, this instruction can be applied to new <cat-toy> images to achieve the same impressionist painting style, without losing the fine details of the test image (Figure 1b, Row 1).

One might suggest an alternative approach to image editing using Textual Inversion, which involves learning two tokens: one for the object and another for the style (e.g., "Painting of <cat-toy> in the style of <watercolor-portraits>''). Figure 1a (Row 2) shows the results of this approach. As can be seen, Textual Inversion still often introduces significant changes that deviate from the original input image. Thus, Textual Inversion is not suitable for accurate image editing.

B Additional Qualitative Comparisons

In the attached HTML file (index.html), we provide additional comprehensive qualitative comparisons for our method versus InstructPix2Pix [1] and SDEdit [4].

We also present qualitative comparisons with other state-of-the-art text-conditioned image editing methods, Imagic [3] and Null-text Inversion [5] (Figure 2). These methods can generate outputs based on given text prompts, such as "A watercolor painting of a cat" (Row 3). However, the outputs often do not match the given reference. The text prompts can also be ambiguous and result in unsatisfactory outputs, as illustrated by the case of "A character in a Pixar movie" (Row 1).

Another challenge is the inconsistency of text-conditioned models, where the same text prompt can produce different outputs for different test images. For example, the text prompt "A frozen waterfall" (Row 6) generates different water colors (blue vs. white) when applied to different test images (Before-and-after pair is from [5]). Our method is more consistent in this case, as the learned instruction might have learned the water color.

C Implementation Details

C.1 Optimization settings

We use the pretrained clip-vit-large-patch14 as the CLIP Encoder in our approach. For instruction initialization [7], we set the caption length for after image to 10 tokens. However, this specific caption length does not affect the optimization algorithm. We can optimize initialization instructions of varying lengths (up to 77 tokens). It takes roughly 7 minutes to optimize for one edit, and 4 seconds to apply the learned instruction to new images.

Specifically, during the optimization process, we freeze the tokens representing the start of text (<|startoftext|>), end of text (<|endoftext|>), and all padding tokens after end of text (<|endoftext|>). We only update the tokens inside the text prompt, called <ins> (between <|startoftext|> and <|endoftext|>) (Figure 3a).

C.2 Instruction Concatenation

We only optimize a fixed number of tokens, so we have the flexibility to concatenate additional information to the learned instruction during inference (Figure 3b). This allows us to achieve more fine-grained control over the resulting images. Figure 4 shows qualitative results of this approach. Users can input extra information to combine or guide the learned instruction according to their preferences.

In the first example, we transform "cat" $\rightarrow$ "watercolor cat". We demonstrate how concatenating extra information to the learned instruction enables both image editing (changing to a watercolor style) and

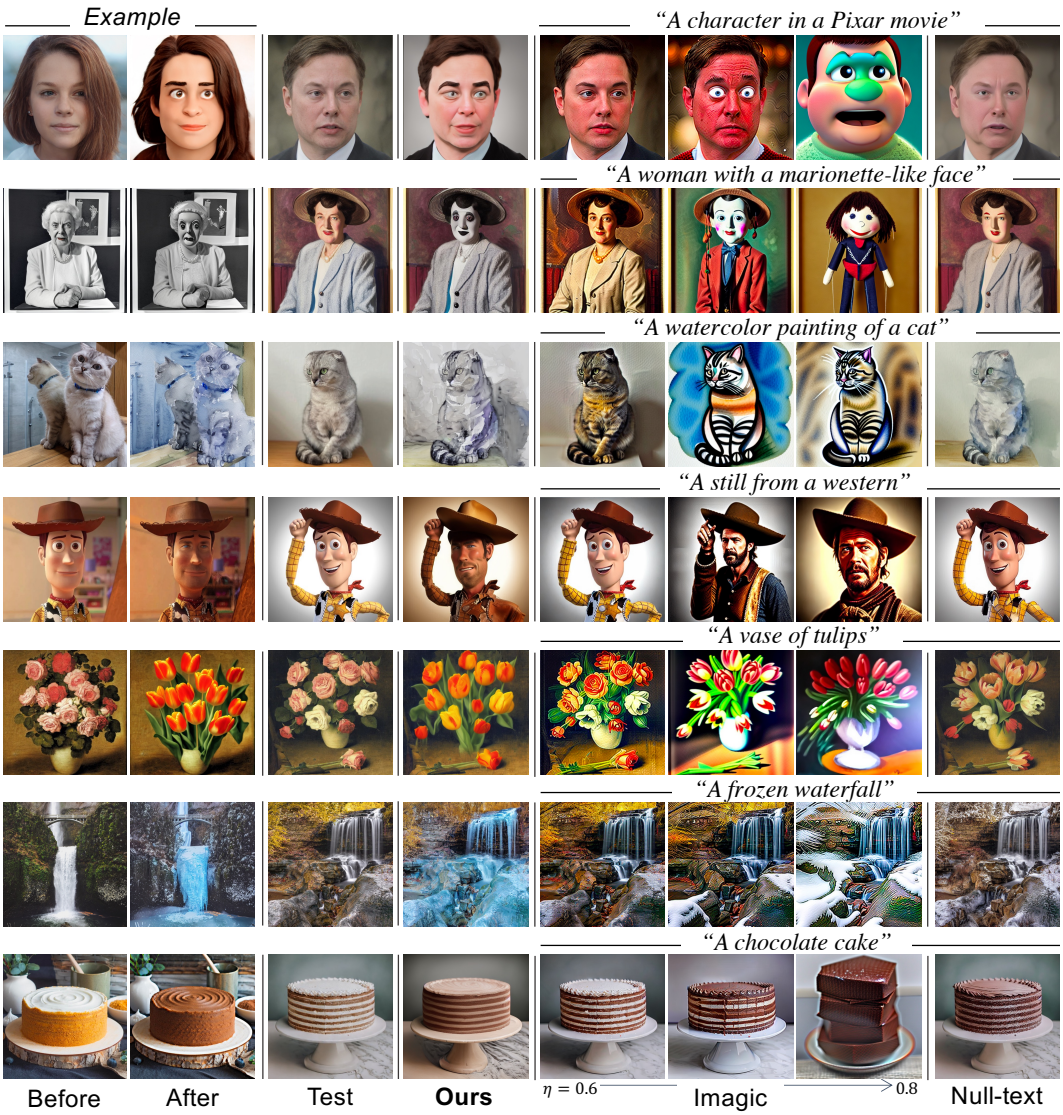

Figure 2: **Additional qualitative comparisons** to Imagic [3] and Null-text Inversion [5]. Imagic and Null-text Inversion fail to match the reference image as they perform edits based on ambiguous text prompts (Row 1-4); or exhibit inconsistency in producing outputs for the same prompt across test images (Row 6). In contrast, our method produces visually closer edited images to the before-and-after pair while demonstrating improved consistency by using the learned instructions.

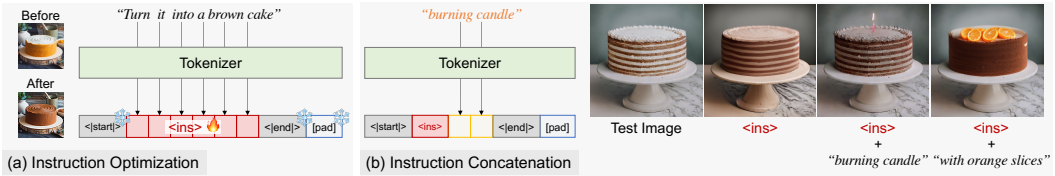

Figure 3: **Implementation details**. (a) Instruction Optimization: We only optimize a part of the instruction embedding, called `<ins>`. (b) Instruction Concatenation: During test time, we can add extra information into the learned instruction `<ins>` to further guide the edit.

domain translation (e.g., "cat" → "tiger"). The painting style is consistent with the before-and-after images, while the domain translation corresponds to the additional information provided by the user.

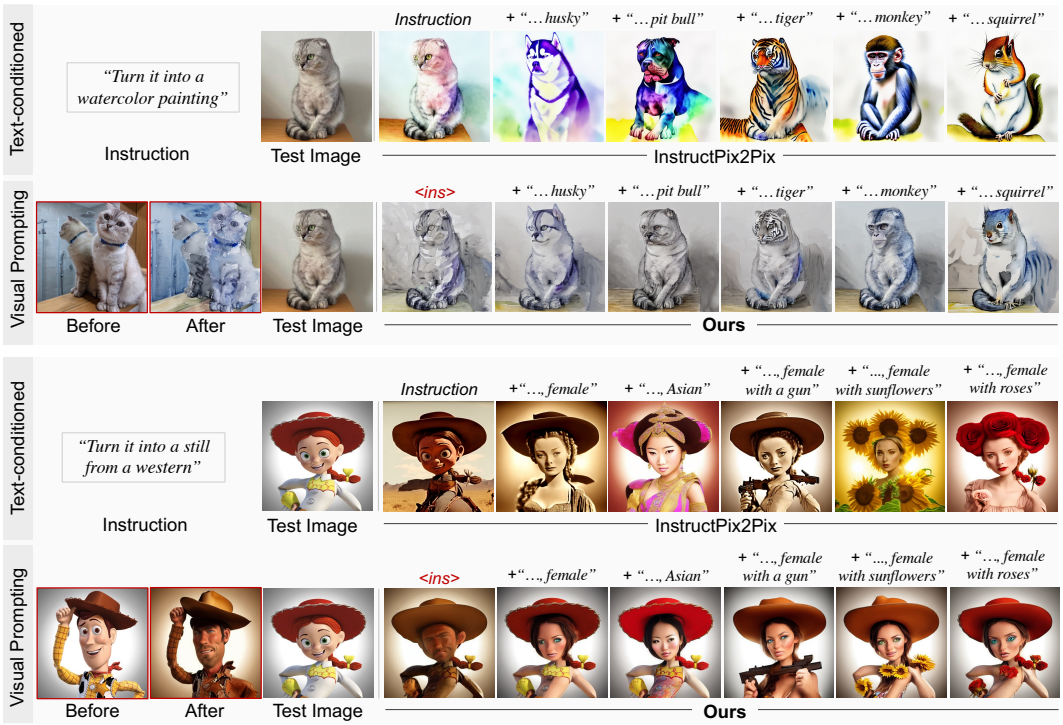

Figure 4: **Instruction concatenation**. We can concatenate extra information into the learned instruction `<ins>` to navigate the edit. (Zoom in for details).

Applying InstructPix2Pix [1] often does not yield satisfactory results, as the painting style differs from the reference image.

In the second example, we further illustrate how concatenating extra information can help guide the learned instruction. We initially learned a male character transformation from a cartoon image. However, applying this learned instruction to a new female test image results in a male bias in the output. By adding extra information, we can navigate the edit to overcome this bias. We can also add extra details such as sunflowers or guns to adjust the learned instruction accordingly. These modifications ensure consistent outputs that are aligned with the original learned instruction. Applying InstructPix2Pix, again, tends to produce more varied outputs that are inconsistent with the user's intention.

Photo Attribution

- Elsa (Human): reddit.com/r/Frozen
- Disney characters: princess.disney.com
- Toy Story characters: toystory.disney.com
- Toonify faces: toonify.photos
- Girl with a Pearl Earring: wikipedia/girl-with-a-pearl-earring
- Mona Lisa: wikipedia/mona-lisa
- The Princesse de Broglie: wikipedia/Princesse-de-Broglie
- Self-portrait in a Straw Hat: wikipedia/self-portrait-in-a-straw-hat
- `<cat-toy>` and `<watercolor-portraits>` concept: huggingface.co/sd-concepts-library
- Gnochi cat, waterfall, and cake images are from Imagic [3] and Null-text Inversion [5].

References

[1] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *arXiv*, 2023.

[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. In *arXiv*, 2022.

[3] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Conference on Computer Vision and Pattern Recognition 2023*, 2023.

[4] 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 *arXiv*, 2022.

[5] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *arXiv*, 2022.

[6] 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 *arXiv*, 2022.

[7] Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. In *arXiv*, 2023.