SPICE: A Synergistic, Precise, Iterative, and Customizable Image Editing Workflow

Kenan Tang*
University of California, Santa Barbara
kenantang@ucsb.edu

Yanhong Li*
Allen Institute for AI
yanhongl@allenai.org

Yao Qin
University of California, Santa Barbara
yaoqin@ucsb.edu

Abstract

Prompt-based models have demonstrated impressive prompt-following capability at image editing tasks. However, the models still struggle with following detailed editing prompts or performing local edits. Specifically, global image quality often deteriorates immediately after a single editing step. To address these challenges, we introduce SPICE, a *training-free* workflow that accepts arbitrary resolutions and aspect ratios, accurately follows user requirements, and consistently improves image quality during more than 100 editing steps, while keeping the unedited regions intact. By synergizing the strengths of a base diffusion model and a Canny edge ControlNet model, SPICE robustly handles free-form editing instructions from the user. On a challenging realistic image-editing dataset, SPICE quantitatively outperforms state-of-the-art baselines and is consistently preferred by human annotators. We release the workflow implementation for popular diffusion model Web UIs to support further research and artistic exploration.[1]

1 Introduction

Image editing is the task of changing the content of an image according to a user's requirements. An example is adding an apple to a specific location on an image (Figure 1).[2] A powerful image editing tool is vital for many applications from creative design to scientific research, including photo editing [17], video editing [6], data augmentation [9], and benchmark construction [26].

Existing vision-language models have achieved initial success on image editing [3, 29]. Such models take in an original image and the user's editing prompt as the input and output the edited image, sometimes with a binary mask as additional input [25, 32]. However, for advanced artistic purposes that require more than one editing step, existing methods are disqualified by the following limitations. First, pixels outside the mask deteriorate after editing. Second, the user cannot specify the precise size and location of an added object by the mask. Third, models struggle with unusual editing tasks, such as adding a backpack to a bench facing away from the viewer (Figure 4). Taken together, these limitations prohibit iterative editing, as image degradation inevitably accumulates.[3]

*Equal contribution

[1] https://github.com/kenantang/spice

[2] All figures are provided in high resolution. Readers are encouraged to zoom in to examine the details.

[3] https://www.reddit.com/r/ChatGPT/comments/1kbj71z/i_tried_the_create_the_exact_replica_of_this/

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: The First Workshop on Generative and Protective AI for Content Creation.

Figure 1: **SPICE enables a user to edit the image exactly as they want, and image details outside the edited region are strictly intact after editing.** More details can be found in Appendix A.

To overcome these limitations, we propose **SPICE**, a *training-free* workflow consisting of 3 steps, namely mask generation, color and edge hint generation, and two-stage denoising (Figure 3).

First, in the mask generation step, the user provides a mask to define the editing region and the context size, localizing the modifications while using essential contextual information from the original image. SPICE strictly constrains the editing region to the user-provided mask, preventing deterioration.

Second, by accepting color and edge hints as a hinted image, SPICE allows the user to provide arbitrarily detailed or simplified image-space information to the model. This resolves the issue that textual prompts cannot specify precise sizes and locations [3, 29].

Finally, SPICE uses a two-stage denoising process, where a Canny edge ControlNet model [30] integrates image-space hints in the *early* denoising steps, and a base diffusion model refines and diversifies the output in the *later* steps. By synergizing the complementary strengths of both models, the two-stage denoising process enables *precise* and *customizable* editing, in which the model faithfully follows the user's requirements on properties of the edited object. Beyond image editing, the superior faithfulness also helps users to overcome fundamental limitations of AI-generated artwork, such as predominantly portraying a single character or repeatedly erring on details.

By overcoming the limitations, SPICE consistently achieves success in iterative editing tasks of *more than 100 editing steps* (Appendix C.3). For the first time, our workflow enables scaling test-time compute [24] in image generation, where the user can decide how each unit of additional compute time contributes to the final image quality. This user-friendly design distinguishes our workflow from contemporary test-time scaling approaches that rely on automatic search algorithms [16, 23].

Despite the strong capability, our workflow can be easily integrated into all popular diffusion model Web UIs, such as Stable Diffusion Web UI Automatic1111, ComfyUI, and Stable Diffusion Web UI Forge. Unlike existing tools such as ADetailer[4] or Regional Prompter,[5] which offer an overwhelming number of hyperparameters that can be difficult to navigate, SPICE provides a much smaller set of hyperparameters for ease-of-use. Moreover, SPICE introduces minimal computational overhead in each editing step, making editing as efficient as generating an image from text with the same model and sampling hyperparameters. This extremely low overhead allows SPICE to run on consumer GPUs (e.g., a single NVIDIA GeForce RTX 4090), unlocking iterative editing for more users. Combining strong capabilities with low implementation and computational cost, our workflow provides a powerful yet accessible tool for researchers and non-researchers alike.

[4] `https://github.com/Bing-su/adetailer`

[5] `https://github.com/hako-mikan/sd-webui-regional-prompter`

(a) DALL·E 3 (b) GPT-4o (c) Ours

Figure 2: **SPICE can generate content that DALL·E 3 and GPT-4o cannot.** Two examples are a rabbit with 4 ears and a violin without a bridge. For individual objects, DALL·E 3 fails even after the user asks the model to edit the errors multiple times. For combined objects, GPT-4o cannot generate all objects correctly at once. However, SPICE can reliably generate these challenging objects.

2 Methods and Results

We defer details of our methods to Appendix B and full evaluation results to Appendix C. We comprehensively evaluate SPICE on challenging editing tasks. Specifically, we stress test the three features of SPICE, namely precise (Appendix C.2), iterative (Appendix C.3), and customizable editing (Appendix C.4). We also provide ablation studies to justify each step in SPICE (Appendix G).

Here, we highlight the results that demonstrate the unique strengths of SPICE over state-of-the-art baselines. Prompt-based image generation or editing models frequently fail to fulfill their promise that users can create content according to their own will. Users are frustrated for two main reasons.

First, prompts are hard to design and cannot precisely instruct the model to produce complex and unusual outputs (Figure 2(a)). A user assumes that the model can follow the prompt, but the model always fails to interpret the prompt as humans do, even after the user repeatedly corrects the model.

Second, the model's randomness makes complex images costly—and even impossible—to generate. For example, if an image must include 10 objects and each is generated correctly only 50% of the time, the chance of getting all 10 right drops below 0.1%. In practice, object-level correctness rate is even lower and scene complexity higher, so achieving perfect results becomes impossible (Figure 2(b)).

With the high customizability of SPICE, instead of desperately engineering prompts, users can effortlessly generate compositions and objects that are impossible for DALL·E 3 and GPT-4o (Figure 2, see full prompt in Appendix C.4).[6] Hence, SPICE creates a new paradigm for human-model collaboration, where users can fully realize their creative vision, without ever needing to compromise to the models' lack of prompt compliance. More instructions and examples of iterative and customizable image generation and editing can be found in Appendix A, Appendix K, and Appendix M.

3 Conclusion

Existing prompt-based image editing models fail in performing local edits, working under different resolutions, following user instructions, and maintaining image quality during multiple editing steps. We propose SPICE, a training-free workflow that addresses all these challenges. We release the workflow to facilitate future research and artistic exploration.

[6]Figure 2(c) has been accepted to CVPR AI Art Gallery 2025. `https://thecvf-art.com/project/compositionality-and-parts/`

Acknowledgments

We thank Anthony Yang for his assistance in preparing the materials presented in Appendix K.

References

[1] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

[2] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. FLUX.1 Kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

[3] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.

[4] Colin Conwell, Rupert Tawiah-Quashie, and Tomer Ullman. Relations, negations, and numbers: Looking for logic in generative text-to-image models. *arXiv preprint arXiv:2411.17066*, 2024.

[5] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10850–10869, 2023.

[6] Xiang Fan, Anand Bhattad, and Ranjay Krishna. Videoshop: Localized semantic video editing with noise-extrapolated diffusion inversion. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision – ECCV 2024*, pages 232–250, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73254-6.

[7] Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Yang Wang, Yinfei Yang, and Zhe Gan. Guiding instruction-based image editing via multimodal large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=S1RKWSyZ2Y`.

[8] Yuying Ge, Sijie Zhao, Chen Li, Yixiao Ge, and Ying Shan. Seed-data-edit technical report: A hybrid dataset for instructional image editing, 2024. URL `https://arxiv.org/abs/2405.04007`.

[9] Yusuke Hirota, Jerone Andrews, Dora Zhao, Orestis Papakyriakopoulos, Apostolos Modas, Yuta Nakashima, and Alice Xiang. Resampled datasets are not enough: Mitigating societal bias beyond single attributes. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8249–8267, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.471. URL `https://aclanthology.org/2024.emnlp-main.471/`.

[10] Yi Huang, Jiancheng Huang, Yifan Liu, Mingfu Yan, Jiaxi Lv, Jianzhuang Liu, Wei Xiong, He Zhang, Shifeng Chen, and Liangliang Cao. Diffusion model-based image editing: A survey. *arXiv preprint arXiv:2402.17525*, 2024.

[11] Mude Hui, Siwei Yang, Bingchen Zhao, Yichun Shi, Heng Wang, Peng Wang, Yuyin Zhou, and Cihang Xie. Hq-edit: A high-quality dataset for instruction-based image editing, 2024. URL `https://arxiv.org/abs/2404.09990`.

[12] Eran Levin and Ohad Fried. Differential diffusion: Giving each pixel its strength. *arXiv preprint arXiv:2306.00950*, 2023.

[13] Haofeng Liu, Chenshu Xu, Yifei Yang, Lihua Zeng, and Shengfeng He. Drag your noise: Interactive point-based editing via diffusion semantic propagation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6743–6752, 2024.

[14] Zichen Liu, Yue Yu, Hao Ouyang, Qiuyu Wang, Ka Leong Cheng, Wen Wang, Zhiheng Liu, Qifeng Chen, and Yujun Shen. Magicquill: An intelligent interactive image editing system. *arXiv preprint arXiv:2411.09703*, 2024.

[15] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022.

[16] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025.

[17] Roey Mechrez, Eli Shechtman, and Lihi Zelnik-Manor. Saliency driven image manipulation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1368–1376. IEEE, 2018.

[18] 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. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=di52zR8xgf`.

[19] 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*, pages 8748–8763. PMLR, 2021.

[20] Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu edit: Precise image editing via recognition and generation tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8871–8879, 2024.

[21] Yichun Shi, Peng Wang, and Weilin Huang. Seededit: Align image re-generation to image editing, 2024. URL `https://arxiv.org/abs/2411.06686`.

[22] Yujun Shi, Chuhui Xue, Jun Hao Liew, Jiachun Pan, Hanshu Yan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8839–8849, 2024.

[23] Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025.

[24] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[25] Su Wang, Chitwan Saharia, Ceslee Montgomery, Jordi Pont-Tuset, Shai Noy, Stefano Pellegrini, Yasumasa Onoe, Sarah Laszlo, David J Fleet, Radu Soricut, et al. Imagen editor and editbench: Advancing and evaluating text-guided image inpainting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18359–18369, 2023.

[26] Xiyang Wu, Tianrui Guan, Dianqi Li, Shuaiyi Huang, Xiaoyu Liu, Xijun Wang, Ruiqi Xian, Abhinav Shrivastava, Furong Huang, Jordan Lee Boyd-Graber, Tianyi Zhou, and Dinesh Manocha. AutoHallusion: Automatic generation of hallucination benchmarks for vision-language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8395–8419, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.493. URL `https://aclanthology.org/2024.findings-emnlp.493/`.

[27] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

[28] Yifan Yang, Houwen Peng, Yifei Shen, Yuqing Yang, Han Hu, Lili Qiu, Hideki Koike, et al. Imagebrush: Learning visual in-context instructions for exemplar-based image manipulation. *Advances in Neural Information Processing Systems*, 36, 2024.

[29] Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. Magicbrush: A manually annotated dataset for instruction-guided image editing. *Advances in Neural Information Processing Systems*, 36, 2024.

[30] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[31] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Scaling in-the-wild training for diffusion-based illumination harmonization and editing by imposing consistent light transport. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=u1cQYxRI1H`.

[32] Haozhe Zhao, Xiaojian Ma, Liang Chen, Shuzheng Si, Rujie Wu, Kaikai An, Peiyu Yu, Minjia Zhang, Qing Li, and Baobao Chang. Ultraedit: Instruction-based fine-grained image editing at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL `https://openreview.net/forum?id=9ZDdlgH6O8`.

Table of Contents

A Hints and Prompts of the Fridge Example **8**

B Methods **8**

B.1 Mask Generation 8

B.2 Color and Edge Hint Generation 10

B.3 Two-Stage Denoising 11

C Overview of Results **11**

C.1 Benchmark Results 11

C.2 Precise Editing 13

C.3 Iterative Editing 14

C.4 Customizable Editing 14

D Evaluation Details **16**

E Human Evaluation Details **17**

F Challenging Examples from More Benchmarks **18**

G Ablation Studies **18**

G.1 Necessity of Components 19

G.2 Flux.1 [dev] Fill Failures 20

H Simple Inputs **21**

I Measuring Properties of Generated Objects **21**

J Changing the Random Seed **25**

K Iterative Construction **25**

L Results of Different Art Styles **26**

M Hyperparameter Recommendations **27**

N Comparison with MagicQuill **29**

O Limitations **32**

P Workflow Implementation **32**

Q Societal Impact **33**

A Hints and Prompts of the Fridge Example

In Figure 1, we show that SPICE enables a user to edit the image exactly as they want, and image details outside the edited region are strictly intact after many editing steps. The first row shows the full image of a 3000×2000 resolution. The second row shows a 900×600 region enlarged for better visibility. In this example, a user uses 9 editing steps to perform various edits, including structure change, object removal, object addition, object replacement, text addition, color change, and detail fixes. Steps 7, 8, and 9 together fix the fridge structure. The labels above the first row are the abbreviated version of the true editing instructions. For example, in the "Add a Word" column, the user wants to add the specific word "suspicious" to the white bowl. The prompt is "An open fridge with food in it. A bowl with a word 'suspicious' on it." The result aligns with the user's requirement.

In Table 1, we list the simple hints and prompts that we use to produce results in Figure 1. We also explain how each hint is quickly and easily added to the image by using Photoshop. The explanation describes the difference between the hinted image at Step i and the result image at Step $i-1$. The results demonstrate two advantages of SPICE. On the one hand, SPICE is highly robust to different types of hints. On the other hand, SPICE allows user to effortlessly achieve sophisticated editing results, without the need to optimize complicated hints or prompts. Sometimes, prompts do not even need to be changed over different steps, as SPICE can infer the user's intent from the hints.

B Methods

SPICE is based on inpainting [15], an operation that uses a diffusion model to replace a masked region on an image, conditioned on a prompt that describes the new image. Note that this prompt differs from the editing prompt, which uses a verb (e.g., add or remove) to describe the editing operation [29]. For example, when the editing prompt is "add an apple in the fridge", the description prompt will be "an apple in a fridge". In this section, we introduce the three key steps of our workflow, namely mask generation, color and edge hint generation and two-stage denoising (Figure 3).

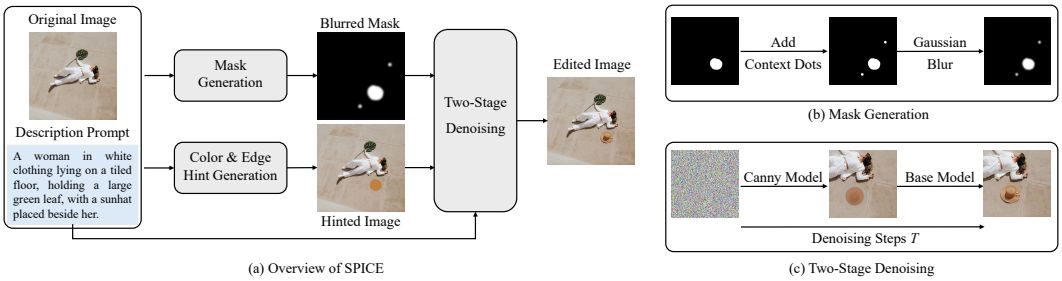


Figure 3: **By sketching a binary mask with context dots and a color & edge hint, users can effortlessly achieve realistic edits with SPICE.** Subfigure (a) shows the overview of our workflow, while Subfigures (b) and (c) show the internal steps. In this example, the user requires a sunhat to be added next to the woman. First, the user sketches both a mask with context dots and a hinted image containing color and edge hints. The mask is automatically blurred after being sketched. Then, during the two-stage denoising step, the Canny and base models perform the early and late denoising steps, respectively. See Figure 17 for more examples of masks and hints.

B.1 Mask Generation

Context Selection. We denote the original image to be edited as $I_T \in [0,1]^{H \times W \times C}$, where T is the index of the editing step, H and W are the image height and width, and $C = 3$ represents the RGB color channels. Other than a prompt p, traditional inpainting methods[7] require the user to provide a binary mask $M \in \{0,1\}^{H \times W}$, where 1 indicates the region to be edited. Given the mask and the prompt, the inpainting operation uses the following steps to generate an edited image. First, a bounding box of the region to be edited is calculated, and the bounding box is extended either

[7] `https://github.com/AUTOMATIC1111/stable-diffusion-webui`

Table 1: **To generate realistic edits, a user only needs to provide simple hints and prompts.** For each step in Figure 1, the image with hints ("Hinted") and the description prompt are shown below. We also explain the hint for each step. The explanation is not part of the input to the model.

Step	Hinted	Result	Description Prompt	Hint Explanation
1			An open fridge with food in it.	The contours of the fridge door and fridge shelves are roughly sketched using the hard round brush.
2			An open fridge with food in it.	The region occupied by the removed bottle is recolored using the hard round brush and the Clone Stamp tool.
3			An open fridge with food in it.	A single red circle is added using the hard round brush.
4			An open fridge with food in it. A beer.	First, the coupe glass is removed using the Clone Stamp tool. Then, an image of Monster Energy is pasted at the same place.
5			An open fridge with food in it. A bowl with a word "suspicious" on it.	Ten black dots corresponding to ten letters of "suspicious" are added on the bowl using the hard round brush.
6			An open fridge with food in it. A purple potato.	The potato is selected by the Quick Selection tool. The hue of the selected region is changed by the Hue/Saturation tool.
7			An open fridge with food in it.	From the image in Step 0, a thin region above the fridge door is selected using the Quick Selection tool and copied to the image.
8			An open fridge with food in it.	From the image in Step 0, a thin region below the fridge door is selected using the Quick Selection tool and copied to the image.
9			An open fridge with food in it.	No change is made.

vertically or horizontally to ensure its aspect ratio matches a user-specified resolution supported by the diffusion model, such as 1216×832 [18]. The user-specified resolution is usually larger than the extended bounding box. Next, pixels on I and M within the extended bounding box are upsampled to the user-specified resolution. Then, the diffusion model generates a new image from latent noise. The generation process is conditioned on the description prompt and the upsampled pixels from I and M. Finally, the output is downsampled to the resolution of the extended bounding box, and the inpainted region on I_T is replaced by the downsampled output, resulting in I_{T+1}.

However, since these traditional methods generate the inpainted region without contextual information outside the extended bounding box, the output often appears unnatural, with inconsistent lighting or color compared to the rest of the image. A naive solution is the "whole image" mode, where the entire image is used as context, and the extended bounding box is not used. However, this introduces two major problems: (1) poor performance on small objects (e.g., deformed fingers or scrambled patterns), and (2) distortion caused by resizing the whole image to a model-supported resolution when aspect ratios of the two differ.

To address these issues, we introduce *context dots*, a pair of dots at opposite corners of the desired bounding box. Context dots ensure that the extended bounding box includes sufficient context for generation. The user directly adds context dots to the original mask, resulting in a context mask $M_{\text{context}} \in \{0, 1\}^{H \times W}$, as shown in Figure 2(b). This simple enhancement offers three key advantages: (1) the user can exclude image areas that interfere with the inpainting process, (2) the user can specify a resolution between that of the inpainted region and that of the full image, balancing local details and global context (Appendix C.4), and (3) context dots minimally affect surrounding pixels, limiting changes to only the desired editing region.

Soft Inpainting. On inpainted images, there is usually an unwanted sharp boundary between the inpainted region and the remaining parts of the image. In more severe cases, an inpainted object can be incomplete. The reason is that even with a context, diffusion models are not robust enough to generate pixels that seamlessly blend with existing ones. To mitigate these artifacts, we adopt Differential Diffusion [12], a method that allows the diffusion model to be conditioned by continuous mask values in [0, 1] during generation. We provide the continuous values as a soft mask $M_{\text{soft}} \in [0, 1]^{H \times W}$ by applying a simple Gaussian blur to M_{context} with a kernel size of a few pixels, as shown in Figure 2(b). This procedure, together with thresholds for blending the original and inpainted image, is named as Soft Inpainting in popular Web UIs. We use Soft Inpainting when it is available in a Web UI or implement our own simplified version (without thresholds) when it is not available.

B.2 Color and Edge Hint Generation

In many existing image editing systems [5, 27, 10], users typically specify desired content through a textual prompt. However, words alone cannot easily describe certain details, such as asymmetric apparel designs or intricate color patterns. Instead, these subtle details can be effectively conveyed with an additional visual hint in the image space. To this end, SPICE allows a user to first create a rough sketch or color layout using standard editing software (e.g., Krita or Adobe Photoshop). The resultant image, denoted $I_{\text{hinted}} \in [0, 1]^{H \times W \times C}$, then replaces the original image as the inpainting input. To incorporate information from this hinted image, a denoising strength hyperparameter in [0, 1] specifies how much the original pixels on the image should be changed.[8] At a moderate denoising strength (e.g., 0.5 to 0.7), the inpainting model produces pixels that remain close to the user's sketched hints, preserving intended colors, shapes, or patterns. Meanwhile, the pixels still diverges enough from the sketch, forming realistic objects in the end.

Note that the user can provide any form of color and edge hints, including sketches, reference images pasted in a collage style, or even another region of the original image (such as using the Clone Stamp Tool in PhotoShop). This flexibility is an improvement over existing methods, such as MagicQuill [14] that uses downsampled 32×32 color blocks for guidance and thus loses high-frequency details. Hence, SPICE allows a user to perform image editing from any point on the human-model collaboration spectrum. At one end, the user fully edit the image by drawing out every detail, without the help of diffusion models. At the other end, the user fully delegates the model to

[8] The hyperparameter does not work as a simple linear blending coefficient at the post-processing stage, and its exact implementation differs for various algorithms. To avoid ambiguity, we refrain from providing a general equation here. Interested readers can refer to the source code of popular Web UIs for the equations.

edit the image. Usually, the user can get decent editing results by staying on the end where the user input is minimal (Appendix C.1). We will also discuss how a small set of hyperparameters enable the user to move freely along this spectrum (Appendix C.4).

B.3 Two-Stage Denoising

Image editing methods [15] typically rely on a single diffusion model to perform all denoising steps when generating an image from latent noise. However, different diffusion models have complementary strengths. On the one hand, a general-purpose text-to-image base model, such as Flux.1 [dev],[9] excels at generating rich variations in its output but can only be conditioned on textual prompts. On the other hand, Flux.1 [dev] Canny,[10] a model derived from Flux.1 [dev] that contains a Canny edge ControlNet (i.e., a Canny model), can be conditioned on Canny edge information [30] but sacrifices variability.

To synergize the strengths, we propose a two-stage denoising process. Specifically, we use a Canny model f_{Canny} during *early* denoising steps to incorporate image-space hints. Starting from latent noise z_0, the Canny model can condition its generation on the Canny edge information E_{hinted} (extracted from the hinted image I_{hinted}) within the extended bounding box, so that the edge hints can be followed. After the early steps, we get an intermediate latent image

$$z_{\text{Canny}} = f_{\text{Canny}}(I_{\text{hinted}}, p, E_{\text{hinted}}, M_{\text{soft}}, z_0). \tag{1}$$

In the *late* denoising steps, we use a base model f_{base} to generate diverse content from z_{Canny}, achieving realism and sophistication despite the simplicity of color and edge hints. This can be formulated as

$$z_{\text{base}} = f_{\text{base}}(I_{\text{hinted}}, p, M_{\text{soft}}, z_{\text{Canny}}). \tag{2}$$

The final latent image z_{base} will be decoded into the edited RGB image. This two-stage denoising process ensures that the denoising process benefits from the strengths of both models. Meanwhile, by simply adjusting the proportion of denoising steps assigned to each model, users can intuitively balance variability and controllability.

Empirically, we find that for a wide range of image editing tasks, Canny Edge ControlNet models outperforms other ControlNet variants (e.g., depth or pose) in the first denoising stage. Furthermore, Canny models are more widely available than other ControlNet models. Therefore, we only use Canny models but not other ControlNet models in our workflow.

C Overview of Results

C.1 Benchmark Results

In this section, we evaluate SPICE under various image-editing scenarios. First, we evaluate our workflow on a standard benchmark of single-step editing. Then, we systematically verify the effectiveness of the three features of SPICE, namely precise, iterative, and customizable editing.

Implementation Details. We use InstructPix2Pix (IP2P) [3], IP2P trained on MagicBrush [29], and UltraEdit [32] as baseline methods. We evaluate all models on the EditEval [10] benchmark, using the standard CLIP [19] text-image direction similarity (CLIP$_{dir}$) and CLIP output similarity (CLIP$_{out}$) metrics from the Emu Edit benchmark [20]. More details can be found in Appendix D.

Quantitative and Qualitative Evaluation. Table 2 shows that our workflow achieves the highest scores on the EditEval benchmark. In addition, we compare the edited images by baseline methods and our workflow on all six different editing categories for visual comparison. Figure 4 shows the failure patterns of baseline methods including undesired global color shift of the grass (Object Addition), removing the lens flare outside the masked region (Object Replacement), inability to remove multiple humans from the image (Object Removal), and wrong anatomy of the polar bear (Action Change). In contrast, our workflow maintains global color, keeps details outside the mask untouched, removes objects as requested, and generates animals with correct anatomy.

[9] `https://huggingface.co/black-forest-labs/FLUX.1-dev`

[10] `https://huggingface.co/black-forest-labs/FLUX.1-Canny-dev`

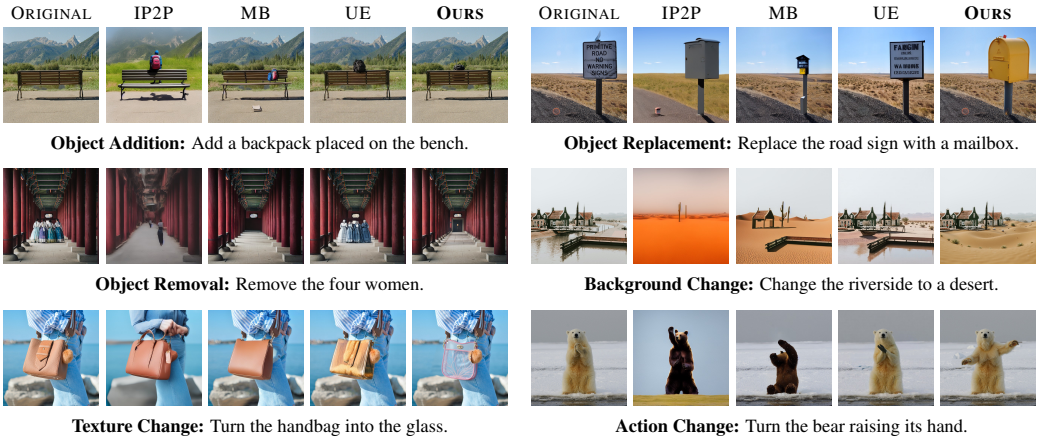


Figure 4: **Our workflow outperforms baseline methods in 6 editing categories from EditEval.** Each group of five images shows an example from an editing task. Each group starts from original image, followed by IP2P, MagicBrush (MB), UltraEdit (UE), and our results.

Table 2: **Our workflow achieves top performance in two quantitative metrics on the EditEval benchmark.** We show mean $\pm$ standard deviation calculated across all images ($n = 126$).

METHOD	CLIP_{dir}	CLIP_{out}
IP2P	0.193 ± 0.120	0.282 ± 0.042
MAGICBRUSH	0.210 ± 0.147	0.302 ± 0.038
ULTRAEDIT	0.193 ± 0.148	0.301 ± 0.039
OURS	$\mathbf{0.221 \pm 0.150}$	$\mathbf{0.305 \pm 0.033}$

Human Study. To provide a comprehensive qualitative evaluation, we further conduct a human evaluation to compare results from different models (more details in Appendix E). Figure 5 shows that our workflow is predominantly preferred. Although at a much lower chance than baseline methods, our workflow does sometimes fail to follow instructions, as indicated by the "both bad" cases. In Appendix C.3, we show how the flexibility of our workflow allows us to overcome this limitation easily.

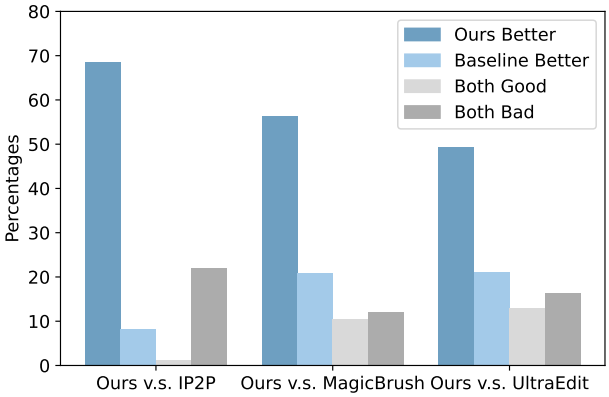


Figure 5: **Our workflow is preferred by the annotators over any baseline method.** In this evaluation, we use only a single editing step and fixed hyperparameters for our workflow, and thus our workflow can still fail ("both bad" cases). In Appendix C.3, we show how relaxing these constraints can dramatically improve results.

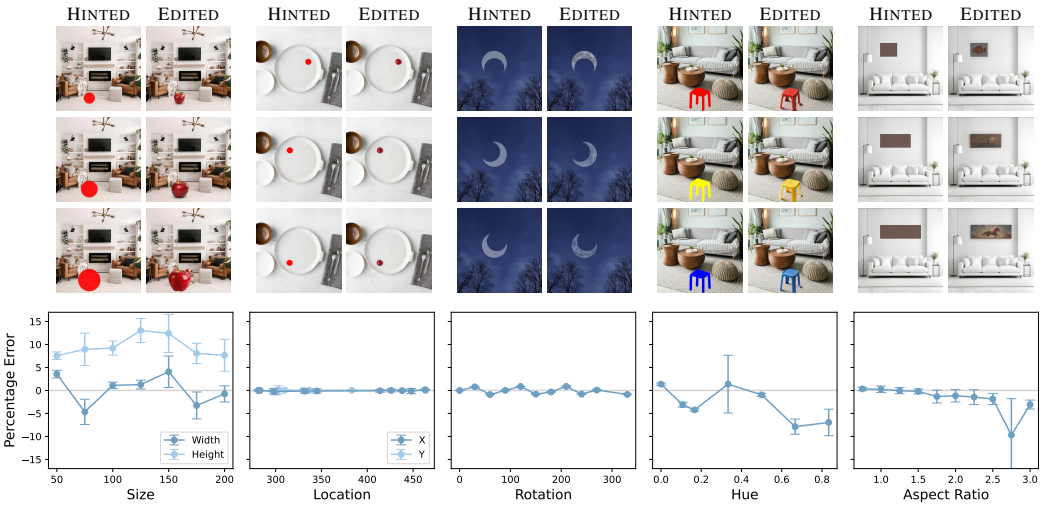


Figure 6: **Our workflow delivers precise and repeatable edits, where the generated objects align with user specifications.** The top images compare the user-specified color hints with the generated objects, and the bottom plots show the percentage errors. On each plot, we show the mean and standard deviation of percentage errors across 10 random seeds. All mean errors are close to 0.

Handling Challenging Edits. In Appendix F, we show that our workflow outperforms baselines on challenging examples from two other popular benchmarks (Emu Edit and MagicBrush). On the challenging examples, our workflow also outperforms GPT-4o, Gemini 2.0 Flash, and SeedEdit [21] (a recent mask-based commercial baseline from Doubao AI).

Ablation Study. While FLUX.1 [dev] is a strong image generation backbone, we demonstrate that our workflow outperforms the backbone model FLUX.1 [dev] by an ablation study. Also, we designed our workflow such that it mitigates many issues of the specialized inpainting model FLUX.1 [dev] Fill.[11] Hence, the better performance of our workflow than baseline methods does not result from simply selecting a stronger backbone. Results can be found in Appendix G.

Minimal Burden on Users. Despite the superior performance, our workflow imposes minimal burden on the users. While the users need to provide both the masks and the hints, the two inputs can be casually sketched, because our workflow is robust to missing details and inaccurate shapes. Examples of excellent editing results from simple user inputs are shown in Appendix H.

Image Editing Benchmarks. Several datasets have been introduced for training and benchmarking image editing models, including EditBench [25], MagicBrush [29], HQ-Edit [11], InstructPix2Pix [3], UltraEdit [32], Seed-Data-Edit [8], and EditEval [10]. While dataset sizes have grown over time, the number of challenging test cases remains limited. Future benchmarks should prioritize difficult structural editing tasks, such as modifying object actions or layouts, rather than focusing primarily on simpler semantic edits like color or texture changes.

C.2 Precise Editing

To quantify the precision of our workflow, we add objects while specifying 5 properties, including size, location, rotation, color, and aspect ratio (Figure 6). To specify the value of each property, we use a hint in the form of a color patch. In the edited image, we measure the property value of the added object and calculate the percentage error against the user-specified property value (Appendix I). We use 10 random seeds to account for generative variability.

Figure 6 shows that the mean of percentage errors for each specified property value is close to 0, with a small standard deviation across random seeds. There are still systematic errors, such as the height

[11] `https://huggingface.co/black-forest-labs/FLUX.1-Fill-dev`

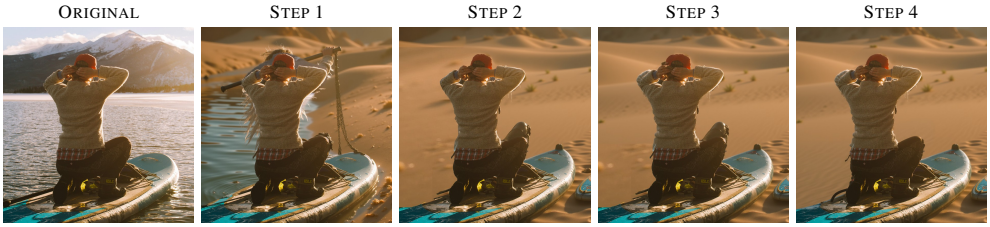


Figure 7: **We use SPICE to recycle failures.** In this example, the editing instruction is "change the lake to a desert". After Step 1, SPICE fails to remove the water from the background. However, 3 more editing steps greatly improve the outcome by removing obvious artifacts. In each step, the mask and hint are redrawn to emphasize different parts of the image for editing, but the description prompt is fixed, imposing low burden on the user.

of the apple being consistently 10% larger than specified. However, for the apple, the height is larger because the apple's stem is unspecified by the hint. Overall, the results confirm that our workflow delivers precise, repeatable edits critical for real-world scenarios.

C.3 Iterative Editing

In this subsection, we consider two iterative editing regimes, one using fewer than 5 editing steps, another more than 100. Multiple editing steps are often necessary, as one single editing or generation step usually fails to align with complicated user requirements (further discussed in Appendix C.4).

In the low-steps regime, if unsatisfied with results from the first editing step, we can improve them using another editing step with our workflow. In Appendix C.1, we identify that our workflow does not work well for a few cases in single-step editing. In these cases, a large proportion of the edited region is satisfactory, leaving only a small proportion to be fixed. For baseline methods, as the image quality degrades after the first editing step, we cannot run a second editing step but can only rerun the model with the same input and different random seeds, and the results are unsatisfactory (Appendix J). However, with our workflow, we can improve the result by recycling the first output (Figure 7).

In the high-steps regime, while many previous methods support multi-step editing, heavy artifacts are generated and propagated in each step, disqualifying these methods for long editing tasks. For example, when GPT-4o is prompted to return the same image to the user,[12] the image quality quickly deteriorates within even a few steps, disqualifying GPT-4o from iterative editing. In contrast, our workflow steadily improves the image after multiple steps. In Figure 8(a), we show a result after 40 diverse editing steps, including outpainting, structural edits, upscaling, relighting, and composition adjustment (forming a heart shape using the sky, the crescent, the claw, and the branches). Full steps are shown in Appendix K. Interestingly, to relight characters according to the environment, we do not need to provide any color hints. We also do not need a specialized relighting model, such as IC-Light [31]. In Figure 8(b), we start with a model-generated image with various errors in shadows, objects, and anatomy. Using the same model as the backbone in our workflow, we fix these errors one by one across more than 100 editing steps. This image is produced by BoleroMix (Pony) v1.41, a checkpoint derived from SDXL. More examples of final edited results in different styles can be found in Appendix L, where around 200 editing steps are performed to achieve the desired level of detail. Each image is created from scratch within 4 hours in one sitting.

C.4 Customizable Editing

Prompt-based image generation or editing models frequently fail to fulfill their promise that users can create content according to their own will. Users are frustrated for two main reasons.

First, prompts are hard to design and often cannot instruct the model to produce complex output. A user assumes that the model follows the prompt, but the model does not interpret the prompt as humans do. One example is that DALL·E 3 [1] in November 2024 still fails to generate a rabbit with

[12] `https://replicateimage.com/examples`

(a) Iteratively constructing an image

(b) Iteratively refining an image

Figure 8: **We use SPICE to consistently improve image quality during a large number of editing steps.** The editing steps can either change the structure or refine the details. The top images are generated using Flux.1 [dev]. The bottom images are generated using BoleroMix (Pony) v1.41.

four ears or a violin without a bridge,[13] as shown in Figure 9(a). Another example is that DALL·E 3 fails to interpret relations or numbers, such as a potato under a spoon or five fish [4].

Second, the innate randomness of the model dramatically increases the generation cost or even prohibits the generation of complicated images. Suppose a user wants to generate an image with 10 different objects at various locations on this image. If each object has a 50% chance to be generated perfectly, the success rate of all objects being perfect is below 0.1%. In real scenarios, the perfection rate for each object will only be lower, and the number of objects larger. Then, generating such an image is practically impossible.

With SPICE, instead of desperately engineering prompts or varying the random seed, users handle these difficulties by tuning 3 hyperparameters, namely context size, denoising strength, and Canny model steps. By slightly tuning the 3 hyperparameters (Appendix M), we are able to generate Figure 9(c), an image with a rabbit of four ears, holding a violin without a bridge with the left hand and a spoon with the right hand. There is a potato under the spoon and five fish in the air.[14] In contrast, DALL·E 3 and GPT-4o both fail to generate the components correctly. The prompt for GPT-4o is "Generate a rabbit with four ears holding a violin without a bridge. The rabbit holds a spoon in her hand. There is a potato under the spoon. There are five fish floating in the air."

[13] `https://cs.nyu.edu/~davise/papers/DALL-E-Parts/PartsNovember24/DALL-E-Parts-November24.html`

[14] Figure 2(c) has been accepted to CVPR AI Art Gallery 2025. `https://thecvf-art.com/project/compositionality-and-parts/`

(a) DALL·E 3 (b) GPT-4o (c) Ours

Figure 9: **SPICE can generate content that DALL·E 3 and GPT-4o cannot.** Two examples are a rabbit with 4 ears and a violin without a bridge. By January 2025, DALL·E 3 fails even after the user points out the error and asks the model to edit the errors multiple times. By May 2025, GPT-4o also cannot generate all elements correctly at the same time. In contrast, our workflow generates all elements correctly in one image, demonstrating superior customizability.

D Evaluation Details

Evaluation Benchmark. We use the second version of the EditEval [10] benchmark to evaluate our workflow and baseline methods. EditEval is a single-step editing benchmark consisting of original images, source captions, target captions, and editing prompts. The original images have resolutions ranging from 1863×1863 to 8742×8742. EditEval covers semantic editing (object addition, object removal, object replacement, and background change), stylistic editing (style change and texture change), and structural editing tasks (action change). Due to the extremely high resolutions of images, many editing tasks in EditEval require challenging fine-grained edits. An example is removing a tiny insect from a bird's beak, which requires editing fewer than 1% of all pixels. EditEval allows a fair comparison of all methods, as none of them were trained on a dataset with the same distribution as EditEval.

Baseline Methods. The baseline methods include InstructPix2Pix (IP2P) [3], IP2P trained on MagicBrush [29], and UltraEdit [32]. For these methods, we use the original editing prompts from EditEval. The inference hyperparameters are set to recommended values. For UltraEdit, we use the mask-based checkpoint and the same binary masks (with context dots) as in our workflow. Hence, UltraEdit similarly benefits from the extra user input, ensuring a fair comparison between our workflow and this strong baseline.

For InstructPix2Pix, we use the model and hyperparameters according the Hugging Face official page.[15] The number of inference steps is set to 10, and the image guidance scale is set to 1.

For MagicBrush, we also refer to the Hugging Face page.[16] Specifically, we use the recommended "recent best checkpoint", which is `MagicBrush-epoch-52-step-4999.ckpt`. We do not provide extra hyperparameters when calling the inference script from the command line, except that we provide a random seed. Hence, all parameters are fixed at their default values, presumably recommended by the model developers.

For UltraEdit, we first use the model version that supports masks.[17] We use the recommended hyperparameter values. We also use the free-form version,[18] since binary masks are not available in Emu Edit and MagicBrush test sets. Since there is not a recommended set of hyperparameters on the

[15] `https://huggingface.co/timbrooks/instruct-pix2pix`
[16] `https://huggingface.co/osunlp/InstructPix2Pix-MagicBrush`
[17] `https://huggingface.co/BleachNick/SD3_UltraEdit_w_mask`
[18] `https://huggingface.co/BleachNick/SD3_UltraEdit_freeform`

official page for the free-form version, we use the same hyperparameters for the free-form version and the masked version.

For our workflow, we use 0.9 denoising strength, 5 Canny model steps, and 25 base model steps. We use Flux.1 [dev] Canny as the Canny model and Flux.1 [dev] as the base model. We also use a LoRA (Midjourney Dreamlike Fantasy FLUX LoRA[19]) with 1.0 strength on the base model to stabilize the output style. We manually draw color patches and masks with a hard round brush, which is a coarse-grained brush that limits the complexity of user input. We also used Photoshop's selection tools whenever necessary, such as the Rectangular Marquee tool and Quick Selection tool. For each original image, we spend less than one minute to do the selection and add the color patch. The color patch opacity is 0.8. To prevent cherry-picking, we fix the random seed at 0. We use target captions from EditEval as description prompts.

Evaluation Metrics. We use the CLIP [19] text-image direction similarity (CLIP_{dir}) and CLIP output similarity (CLIP_{out}) metrics from the Emu Edit benchmark [20]. These two metrics are suitable for reference-image-free evaluation. We do not calculate the L1 distance between edited and original images, as the L1 distance does not monotonically increase with the editing quality (higher distance can simultaneously indicate better global editing performance and worse local editing performance). In EditEval, we exclude the style change task, as the task is better handled by style LoRAs or specialized checkpoints (e.g., Flux.1 [dev] Redux).[20] After excluding 24 images, there are $n = 126$ images remaining.

E Human Evaluation Details

For human evaluation, we adopt a preference setting. An annotator is asked to choose a better image out of a pair or consider both images to be "good" or "bad". We construct an *image pair* by using one image from our workflow and another image from one of the baselines. In total, there are $3 \times 126 = 378$ image pairs for 3 baselines. For each of the 3 subsets (126 image pairs), we ask 3 annotators for evaluation, totaling 9 annotators. For each image pair, the annotator is shown 3 images, including the original image, an edited image from one method, and an edited image from another method. The order of our workflow and the baseline is randomly chosen for each image pair. The order is hidden from the annotators. We show the editing category and the editing prompt, but not the source prompt or the target prompt. Before the annotation process begins, each annotator is given the same evaluation instructions, where we do not specify detailed criteria for preference other than the adherence to the editing instruction.

An example of an image pair the annotator sees is shown in Figure 10. To ensure fairness, we send the same instructions to the annotators. To ensure minimal bias, we do not explain the relative strengths and weaknesses of each method. We do not further communicate with annotators or provide clarifications in written or spoken form, until the annotators finishes the task. The only message we send to annotators is shown below (from "Hi" to "winner").

Hi [NAME],

We are working on a project about using diffusion models to edit images. We would like you to help evaluate the results. This evaluation requires you to choose the better image from a pair. There are a total of 126 pairs, so the evaluation would not take long.

The image pairs can be found at

[LINK TO THE FOLDER CONTAINING IMAGE PAIRS]

Please submit your evaluation using this spreadsheet

[LINK TO THE SPREADSHEET]

Put a 1 in the option you choose. Please feel free to discuss anything you observe with us. Thanks for your help!

The evaluation instruction:

[19] `https://civitai.com/models/679736/midjourney-dreamlike-fantasy-flux-lora`

[20] `https://huggingface.co/black-forest-labs/FLUX.1-Redux-dev`

Please choose from A and B the image that better follows the editing instruction. If the instruction is followed by both A and B, choose the image that looks better. Only choose "both good" or "both bad" if there is no clear winner.

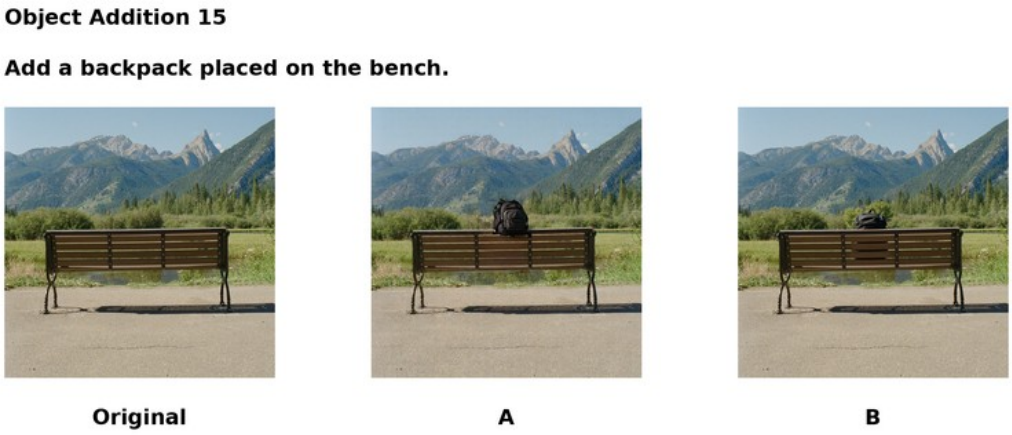


Figure 10: **An example of an image pair that the annotator sees.** The annotator does not know which option comes from which model. While they can see the name of the baseline method they are evaluating, none of the annotators know about the baseline methods before performing annotation.

F Challenging Examples from More Benchmarks

We further compare the performance of our workflow against baseline methods on selected challenging examples. A challenging example is an example where all baseline methods fail. We select challenging examples from the Emu Edit and MagicBrush test sets. For the Emu Edit test set, the baseline methods are Emu Edit, IP2P, MagicBrush, UltraEdit, Doubao SeedEdit, Gemini 2.0 Flash, and GPT-4o (Figure 11 and Figure 12). For the MagicBrush test set, the baseline methods are Reference (the best DALL·E 2 generations), IP2P, MagicBrush, UltraEdit, Doubao SeedEdit, Gemini 2.0 Flash, and GPT-4o (Figure 13 and Figure 14). After we exclude the style change task, both test sets cover 7 different editing categories, as reported in the original papers. We select 2 challenging examples from each category, totaling 14 for each test set. We translate the original prompts into Chinese when using Doubao SeedEdit.

Our workflow qualitatively outperforms strong baselines including Doubao SeedEdit, Gemini 2.0 Flash, and GPT-4o. Notably, our workflow strictly preserves details outside the edited region. In contrast, fine-grained texture of the image corrupt severely after a single step of editing by either Gemini 2.0 Flash or GPT-4o, which is an innate limitation of mask-free methods. Moreover, GPT-4o stretches a square input image into a rectangular image without being prompted, leading to inconsistent resolutions.

We encourage readers with more experience in the baseline methods to independently verify their performance upper-bound.

G Ablation Studies

We qualitatively demonstrate the effect of removing each component from our workflow (Figure 15). Then, we demonstrate that our workflow mitigates many issues of the specialized inpainting model FLUX.1 [dev] Fill (Figure 16). Together, the analysis in this section shows that our workflow improves upon the backbone image generation model.

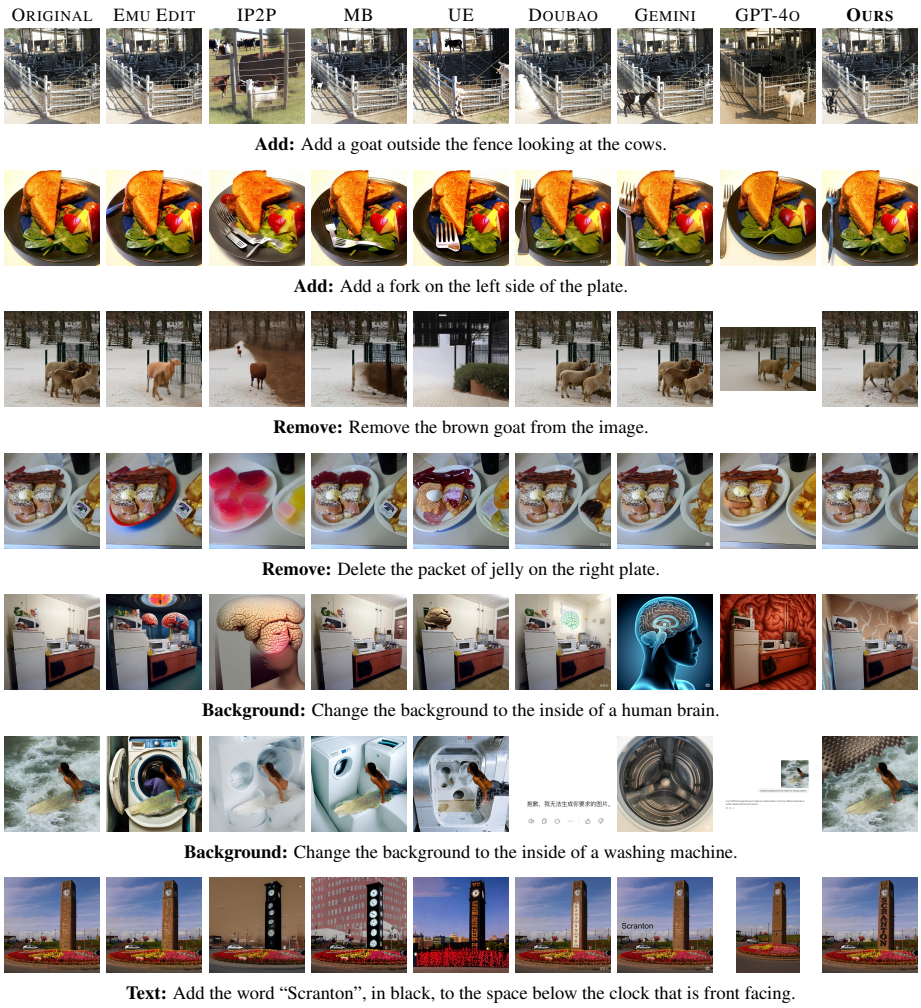


Figure 11: **Our workflow performs the best on the first half of 14 challenging examples from Emu Edit.** Both Doubao AI and GPT-4o refuses to generate an image for the washing machine example, showing an error message instead.

G.1 Necessity of Components

Removing any component from our workflow leads to the following negative consequences (Figure 15).

Context Selection. Without an appropriate context, the generated region might show a different style than the overall image. Even when the style is correct, the inpainted region could still show a very different texture, an artifact that distinguishes the region from surrounding pixels.

Soft Inpainting. Without soft-inpainting, both major and minor artifacts appear. Here, the major artifact is the missing shoes from the doll. The minor artifact is the abrupt change of texture around the mask edges (on the bench backrest).

Color and Edge Hinting. Without color and edge hints, the object sometimes can still be added. However, the size of the object does not follow the user's requirement.

Two-Stage Denoising. Without two-stage denoising, the generated object and the color patch have different sizes.

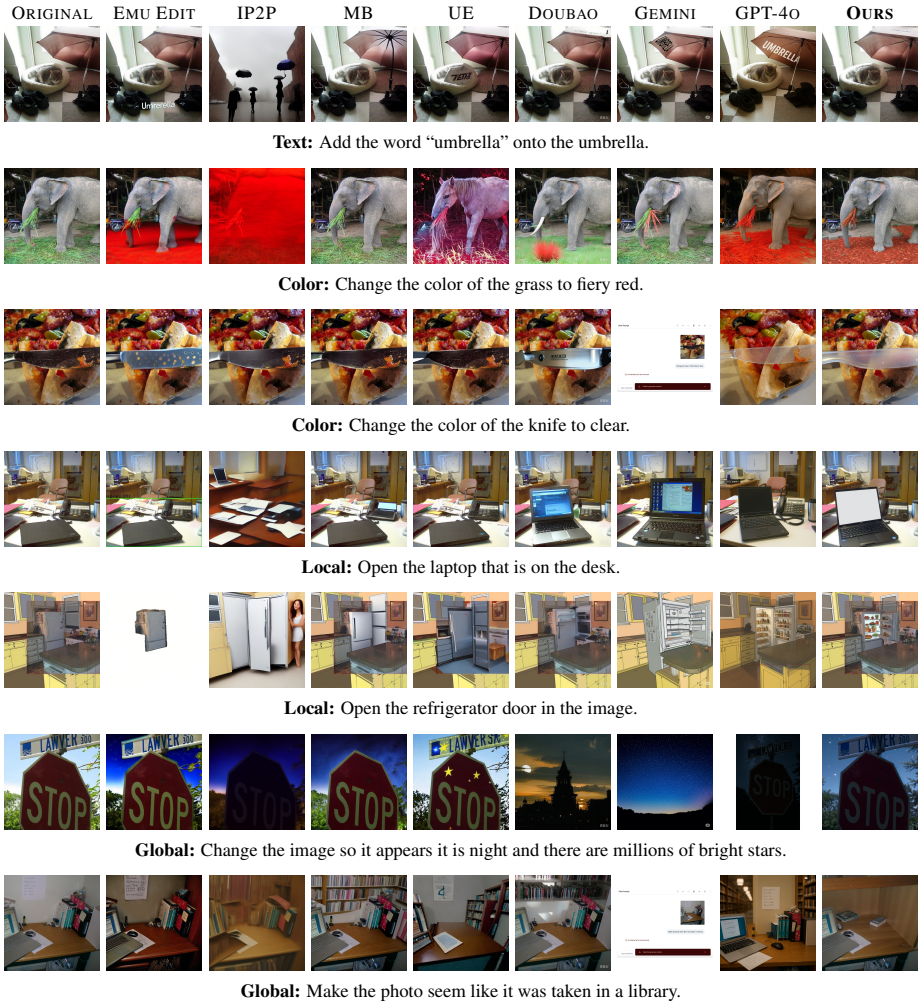


Figure 12: **Our workflow performs the best on the second half of 14 challenging examples from Emu Edit.**

G.2 Flux.1 [dev] Fill Failures

Our workflow mitigates many issues of the Flux.1 [dev] Fill, a specialized inpainting model trained on the same backbone of Flux.1 [dev]. We discuss the issues below and demonstrate them in Figure 16.

Global Editing Failures. While Flux.1 [dev] Fill performs comparably with our method in some local editing tasks, it has a low success rate in global (background) editing. This is probably because Flux.1 [dev] Fill was trained with more images with local edits than ones with global edits. Our method requires no further training, so no additional bias is imposed on either local or global editing.

Color Drifts. Flux.1 [dev] Fill sometimes results in an obvious color drift outside the mask (saturation of the grass color drops), a weakness acknowledged by the official developers.[21] This is because unlike our workflow, Flux.1 [dev] Fill is not strictly a local editing model. By design, our method strictly preserves pixel values outside the mask.

Prompt Bleeding. When multiple objects are mentioned in the prompt, Flux.1 [dev] Fill suffers from the well-known prompt bleeding issue, where characteristics of objects are mixed instead of being independent (a flamingo that looks like a camel). Empirically, by providing color and edge

[21] `https://huggingface.co/black-forest-labs/FLUX.1-Fill-dev#limitations`

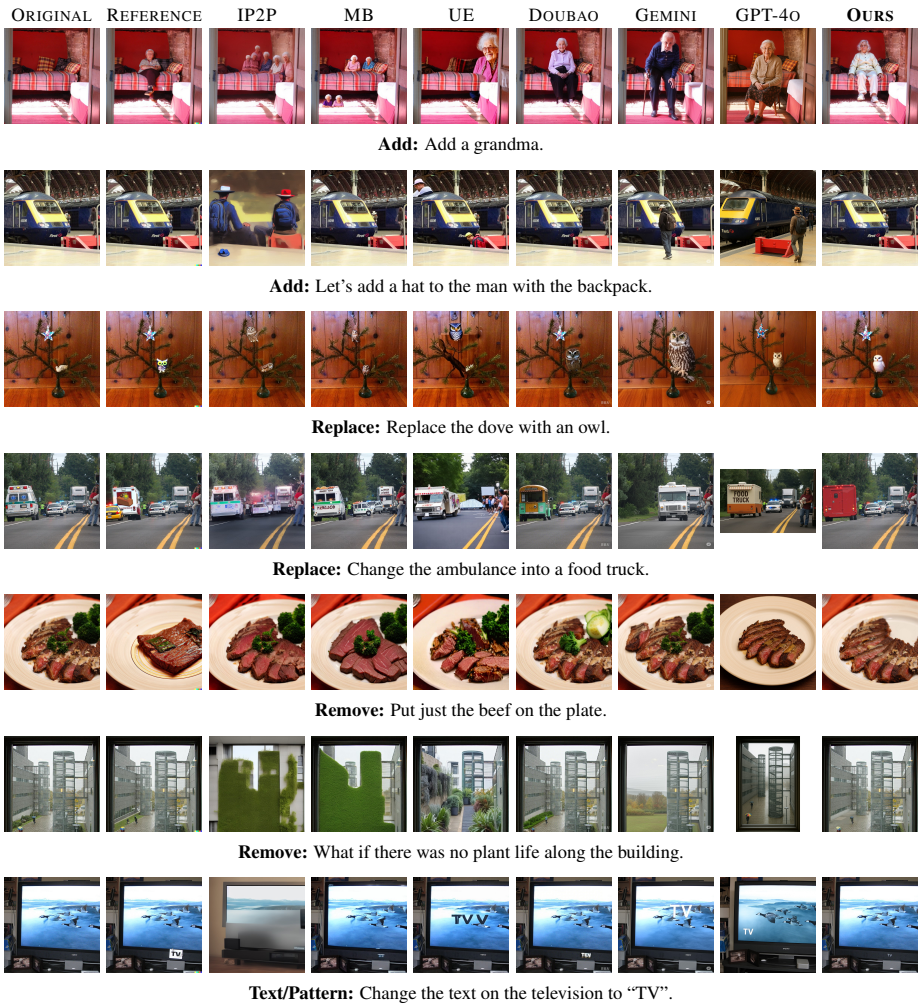


Figure 13: **Our workflow performs the best on the first half of 14 challenging examples from MagicBrush.**

hints, our method effectively handles the prompt bleeding issue, despite using the same prompt containing multiple objects. Moreover, including multiple objects in the prompt improves model's understanding of the context in our workflow, which sometimes leads to even better results.

H Simple Inputs

While our workflow accepts extra inputs from the user, generating the extra inputs poses minimal burden. In Figure 17, we show that simple color hints (HINTED) and simple masks (MASK) can produce high-quality edits.

I Measuring Properties of Generated Objects

To quantify the precision of our workflow, we need to measure the properties of the generated objects. The properties include size, location, rotation, color, and aspect ratio. We use the Language Segment-Anything[22] tool to identify the segmentation mask of a generated object. Here, we use segmentation mask to refer to the pixels that cover the object. We use the default model (`sam2.1_hiera_small`) and settings provided by the developers, which empirically performs well for our purpose. For

[22] `https://github.com/luca-medeiros/lang-segment-anything`

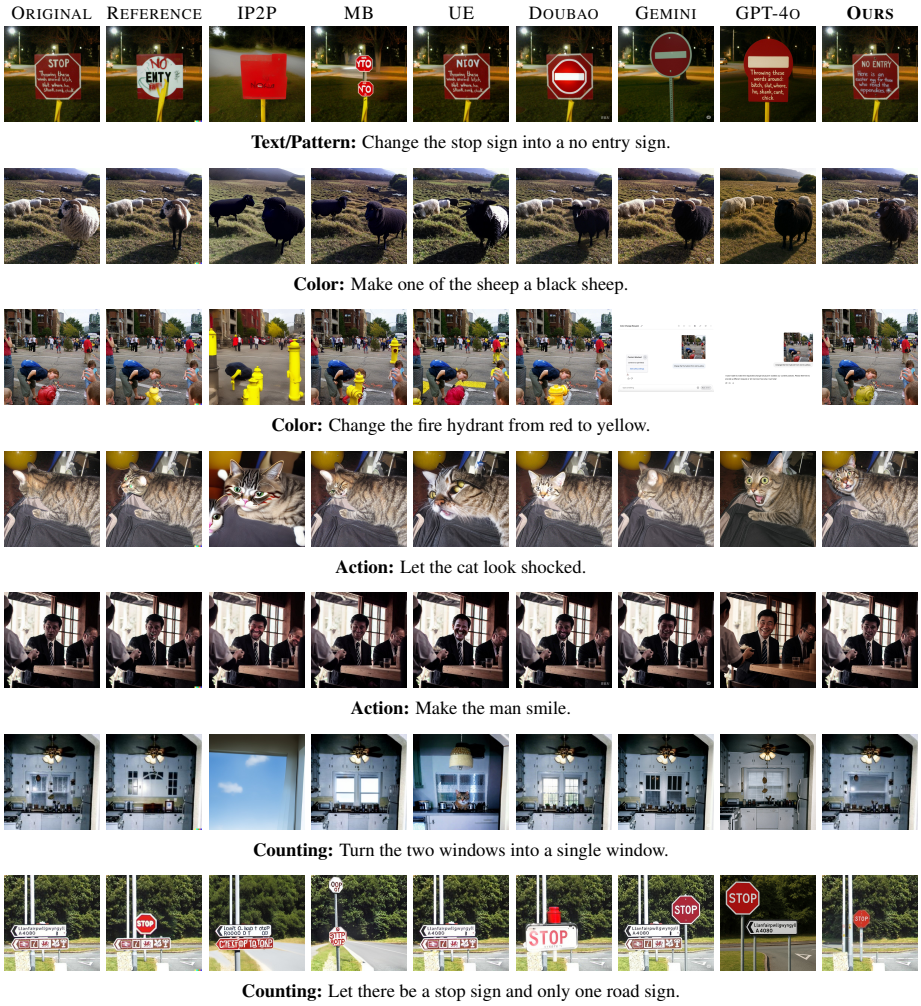


Figure 14: **Our workflow performs the best on the second half of 14 challenging examples from MagicBrush.**

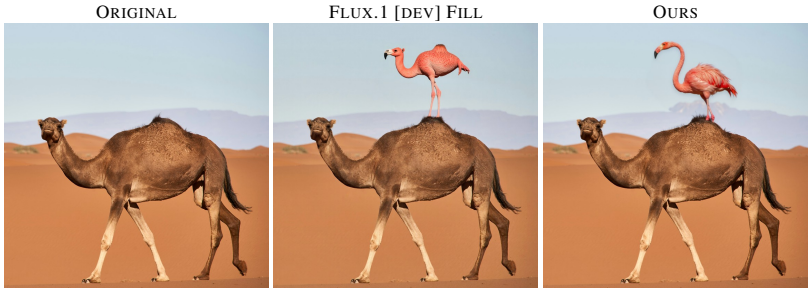
Figure 15: **Ablation studies show that each component of our workflow is necessary.** Removing any one leads to suboptimal quality.

an object, its width and height are defined as the width and height of the bounding box of the segmentation mask. The location is defined as the center of the bounding box. The rotation (for the crescent) is calculated as the direction from the center of mass of the segmentation mask to the center of the bounding box. The color is defined as the average RGB value of pixels on the generated image covered by the segmentation mask. To find a single value to represent the color, we convert the average RGB value into HSV representation and use the hue. The aspect ratio is the width divided by height.

After measuring these properties of the generated object, we calculate a percentage error between the generated property value and specified property value. For size, we vary the diameter of a red circle and compare the height and width of the apple's bounding box with the diameter. For location,

(a) **Global Editing Failures:** A person sits in a bedroom.

(b) **Color Drifts:** Two horses grazing on a grassy field under a clear sky.



(c) **Prompt Bleeding:** A flamingo standing on a camel walking on a desert.

Figure 16: **Our workflow mitigates issues of the strong Flux.1 [dev] Fill model.** While Flux.1 [dev] Fill is a specialized inpainting checkpoint, it suffers intrinsic limitations including failures in global editing, color drifts, and prompt bleeding. In contrast, our training-free workflow is free from these issues. In the captions, we show the description prompt for each editing task. Note that we do not use the editing prompt ("add a flamingo on top of the camel"), because these two methods are both designed to accept description prompts instead of editing prompts.

we vary the center coordinates of a red circle and compare the center coordinates of the cherry's bounding box with the center coordinates of the circle. For rotation, we vary the rotation angle of a crescent-shaped color patch and compare the orientation of the crescent with the orientation of the color patch. For color, we vary the hue of a color patch and compare the average hue of the plastic chair to the hue of the patch. For aspect ratio, we vary the width-to-height ratio of a rectangle and compare the width-to-height ratio of the painting with the width-to-height ratio of the rectangle. The definition of percentage errors are listed below:

$$\text{Percentage Error of Width} = \frac{\text{Generated Width} - \text{Specified Width}}{\text{Specified Width}} \times 100\%,$$

$$\text{Percentage Error of Height} = \frac{\text{Generated Height} - \text{Specified Height}}{\text{Specified Height}} \times 100\%,$$

$$\text{Percentage Error of X} = \frac{\text{Generated X} - \text{Specified X}}{\text{Specified X}} \times 100\%,$$

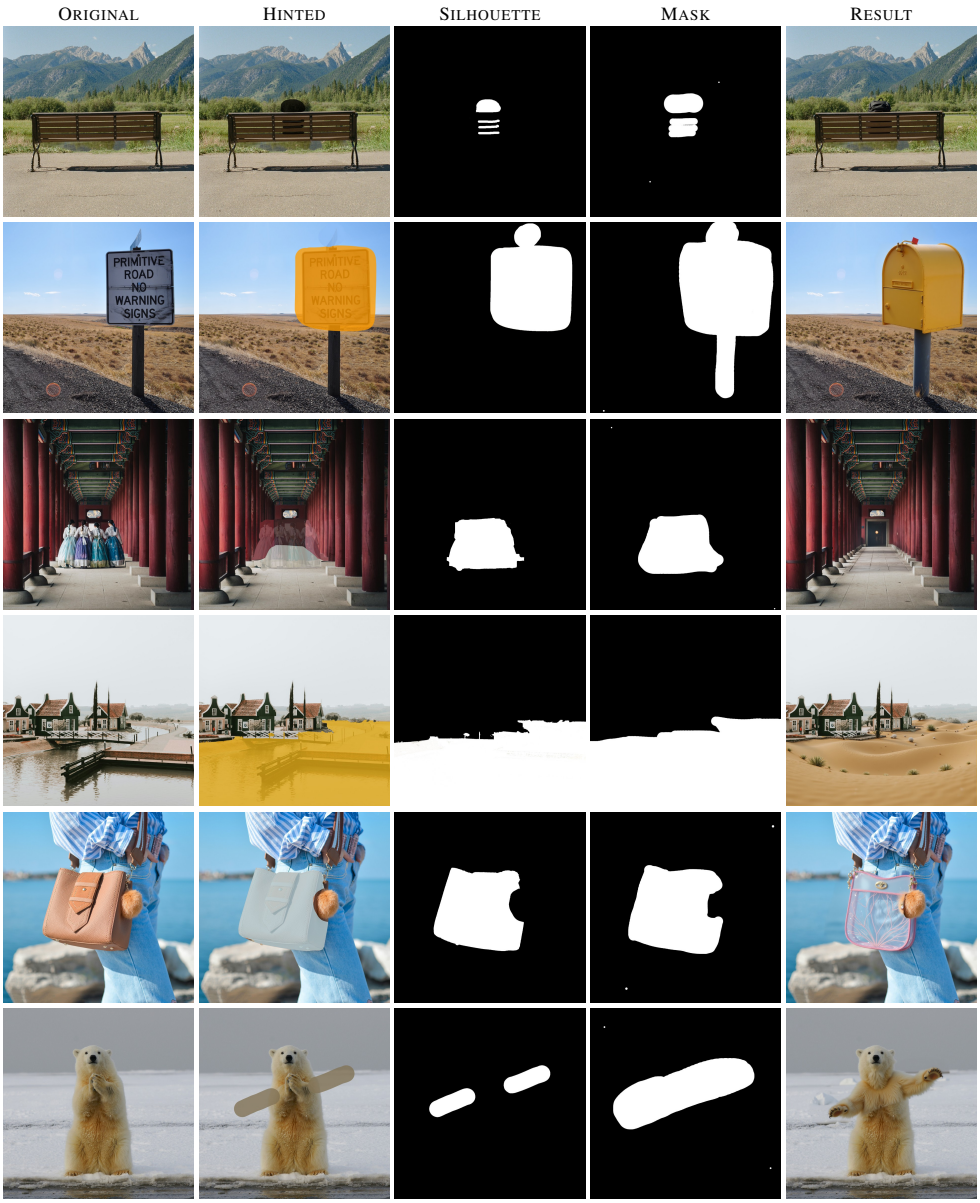


Figure 17: **SPICE can achieve excellent editing results from simple user inputs.** To better visualize the hint, we further show the silhouette of the hint. Silhouettes are not used in the generation process, but are manually extracted after generation for visualization purpose only. Neither the hints nor the masks need to precisely match the desired shape of the edited object.

$$\text{Percentage Error of Y} = \frac{\text{Generated Y} - \text{Specified Y}}{\text{Specified Y}} \times 100\%,$$

$$\text{Percentage Error of Rotation} = \frac{\text{Generated Rotation} - \text{Specified Rotation}}{360^\circ} \times 100\%,$$

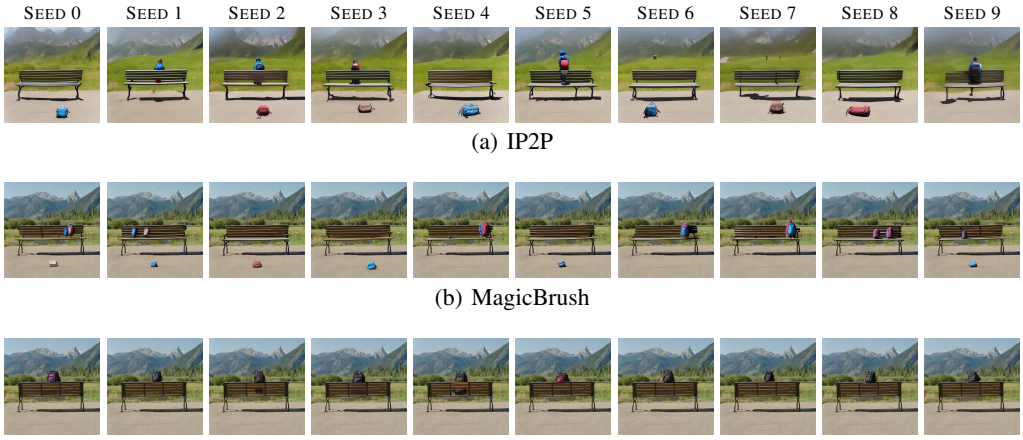
$$\text{Percentage Error of Color} = \frac{\text{Generated Hue} - \text{Specified Hue}}{1.0} \times 100\%,$$

$$\text{Percentage Error of Aspect Ratio} = \frac{\text{Generated Aspect Ratio} - \text{Specified Aspect Ratio}}{\text{Specified Aspect Ratio}} \times 100\%.$$

To account for the periodic nature of rotation and hue, we use 360° for rotation and 1.0 for hue instead of the specified property value.

J Changing the Random Seed

With baseline methods, a user can change the random seed to get multiple results and select the best one. However, even with more compute, baseline models will consistently fail on the same challenging example (Figure 18). In contrast, with our workflow, a user can better utilize the increased compute if the user performs editing step by step.



(a) IP2P

(b) MagicBrush

(c) UltraEdit

Figure 18: **For baseline methods, changing the random seed does not help.** The random seeds are shown on the top. With 10 times the original compute, baseline methods still fail on the challenging editing task from EditEval of adding a bag to the bench. Our workflow succeeds in the first edit and will succeed in fewer than 10 edits should the first edit fail. The original image and our result are in Figure 4.

K Iterative Construction

Figure 23 shows all 40 steps we use to iteratively construct Figure 8(a). Some steps involve subtle but important edits that are best visualized at a high resolution. We will release a video to better illustrate the editing operation done at each step.

We further evaluate iterative editing on recent (October 2025) state-of-the-art models and qualitatively demonstrate how image quality worsens as edits accumulate. The three models we evaluate are **FLUX.1 Kontext [pro]**, **Sora (GPT)**, and **Gemini 2.5 Flash Image (Nano Banana)**. Access to FLUX.1 Kontext [Pro] and Sora was provided by their respective official websites, and access to Gemini 2.5 Flash Image (Nano Banana) was provided via Google AI Studio.[23] For each model, we apply a sequence of 12 iterative edits to the same base image of a man sitting on a chair, wearing a yellow jacket.[24] The editing prompt at each step was: "Change the man's jacket color to x" where x cycles twice through the colors red, orange, yellow, green, blue, and purple, for a total of 12 editing steps. All models failed in this iterative editing task, with image quality dropping dramatically after 12 steps. Below, we further explain and visualize the failure patterns of each model.

FLUX.1 Kontext [pro]. This model is a diffusion-based image editor with strong local inpainting and instruction-following capabilities [2]. However, repeated edits introduce obvious artifacts, as shown in Figure 19.

[23] `https://flux1.ai` and `https://sora.chatgpt.com`

[24] `https://www.pexels.com/photo/man-in-yellow-long-sleeve-shirt-sitting-on-black-chair-smiling-7562191/`

Figure 19: **Noise accumulates in the iterative editing results by FLUX.1 Kontext [pro].** Obvious artifacts appear at around 7 steps.

Sora. Sora preserves global structure well in early steps but changes the aspect ratio from the first step. Moreover, degradation appears as gradual over-smoothing and inconsistency in fine detail, as shown in Figure 20. It also struggles with preserving facial structure as noted in practitioner reports.[25]

Figure 20: **Sora does not preserve details over 12 iterative editing steps.** The background color changes from gray to light blue. The facial structure of the man also changes.

Gemini 2.5 Flash Image. During iterative editing, Gemini 2.5 Flash Image preserves global structure reliably, but texture and details still degrade, as shown in both Figure 21 and Figure 22.

L Results of Different Art Styles

Figure 24 shows two additional results. One is generated with BoleroMix (Pony) v1.41 as the backbone, and the other is generated with the same Flux.1 [dev] as the backbone. These additional results illustrate the flexibility of our workflow in editing and iteratively generating images of different art styles. Note that these are fan-made images, where details may differ from those of the original characters.

[25] `https://davidharris.uk/2025/04/11/kling-vs-runway-vs-sora-vs-lumalabs-a-deep-dive-into-ai-video-generation-platforms/`

Figure 21: **Gemini 2.5 Flash Image fails to maintain character consistency over 12 editing steps.** The background, skin tone, and details on the jacket all change.

M Hyperparameter Recommendations

We first demonstrate the effect of each hyperparameter (Figure 25). Then, we recommend several combination of hyperparameters can help the user reliably generate an image that DALL·E 3 and GPT-4o cannot (Figure 9).

Context Size. During inpainting, a diffusion model generates content consistent with what is already in the context. In Figure 25(a), the user requires another yellow vegetable to be added to the center of the image. The prompt is "yellow vegetables on an orange background". The color hint is the yellow circle in the center. The model's interpretation of the color hint depends on the selected context. When a small context includes none of the other vegetables, the model generates multiple vegetables from the single yellow circle. When a medium context includes corners of the other vegetables, the model generates one vegetable. When a large context includes all other vegetables, the model fails to generate a new vegetable. When the context includes one vegetable either on the top left or the top right corner, the model generates a vegetable with the same direction of shade. While we do not theoretically explain the failure when using large context sizes, we observe it frequently in other editing tasks. We also observe the failure to add objects using the "whole image" mode (Appendix B.1), an observation that motivated our design of context dots.

Denoising Strength. The model balances hinted colors and realistic shadows at a medium denoising strength. In Figure 25(b), the user requires a Rubik's cube with certain face colors to be added to the image. Because the colors cannot be succinctly described by words, the user inserts a reference image as the hint. With a low denoising strength, the colors are preserved, but the shadows are not generated. We observe the opposite with a high denoising strength.

Canny Model Steps. The model balances realistic details and hinted edges at a medium number of Canny model steps. In Figure 25(c), the user requires a crescent to be added to the sky, but the prompt is simply "a moon". With few Canny model steps, the moon looks realistic, but its shape is wrong. We observe the opposite with many Canny model steps.

To generate Figure 9(c), the recommended hyperparameter ranges are shown in Table 3. Note that several consecutive editing steps are necessary in a certain editing region to achieve the shown effect.

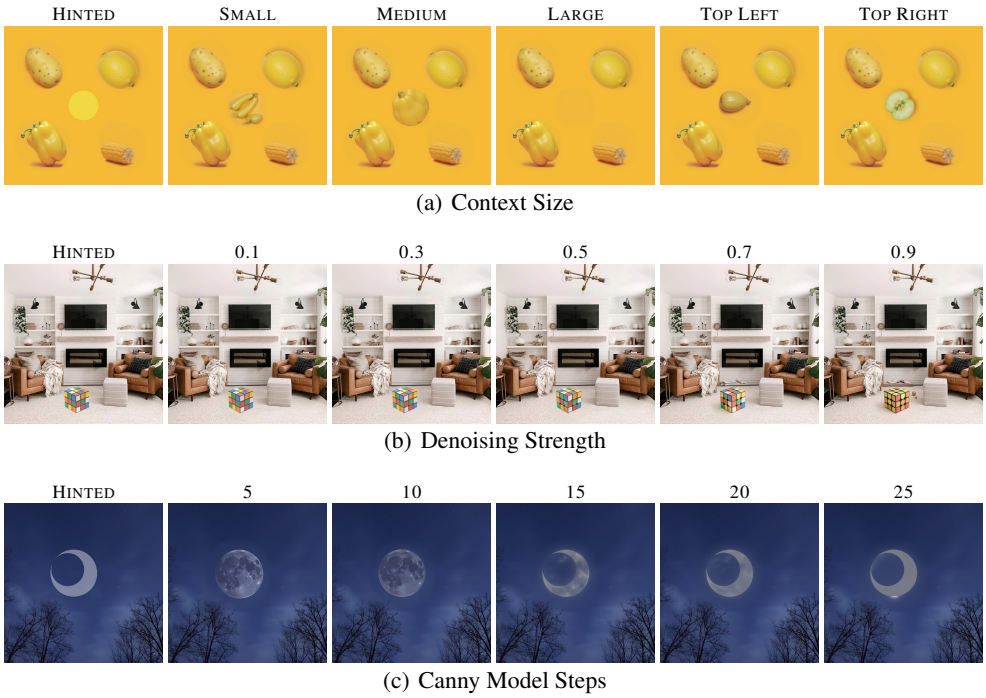
Figure 22: **For clarity, we present a full-page comparison between the initial image and the final output from Gemini 2.5 Flash Image.** The skin tone shifts noticeably, and the background becomes darker and more textured. The buttons on the man's jacket disappear in the edited version, and fine details (eyes, teeth, and fingernails) morph and differ from the original. Additional indentations and fine lines also appear around the mouth.

Figure 23: **We use 40 steps to iteratively construct the image and fix detail errors.** With our workflow, high frequency details do not deteriorate over the steps. With baseline methods, the deterioration happens at the first step.

Figure 24: **Our workflow excels at editing and iteratively generating images of different art styles.** Both images are generated using around 100 to 200 editing steps, within 4 hours in one sitting. The characters are from *Touhou Project* and *Hades 2*. Character LoRAs are not used, so many editing steps are spent on correcting details of the character's clothes. Note that due to the complex composition and high resolution (2048×2048), it is extremely hard, if not impossible, to generate these images with a single text-to-image step.

N Comparison with MagicQuill

Diffusion model-based image editing techniques can be categorized based on the type of inputs used to guide image generation [5, 27, 10]. Most existing methods rely primarily on textual prompts [3, 7], while many also incorporate masks to specify the editing region [25, 32]. Due to the inherent imprecision of both prompts and masks, some methods integrate additional inputs to provide finer control over the generation process [28, 22, 13].



(a) Context Size

(b) Denoising Strength

(c) Canny Model Steps

Figure 25: **Users can achieve desired effects by customizing the value of 3 hyperparameters.** (a) To improve the consistency of edited region with a certain part of the image, the user can cover this part in the context. (b) Lower denoising strength preserves the color, whereas higher denoising strength inserts realistic shadows. (c) Fewer Canny model steps produce high-frequency details, whereas more steps allows the generated image to faithfully follow the hinted shape. Some examples look sub-optimal, because they show the results of only varying one hyperparameter. In practice, users can adjust all three parameters together for better edits.

Table 3: **We recommend different hyperparameter combinations according to specific editing tasks.** Empirically, these combinations have a higher success rate than others.

OBJECT	CONTEXT	DENOISING	CANNY
Ears	Head	Low	High
Bridge	Violin	Moderate	Moderate
Potato	Arm	High	Moderate
Fish	Dress	Moderate	Low

Among these, MagicQuill [14] is most closely related to our workflow. MagicQuill also accepts edge and color information from the user as its input. However, the following differences in design choices lead to a better performance of our method.

Fewer ControlNets. MagicQuill uses three ControlNets, namely inpainting, edge, and color ControlNets. The control branch also needs to be further trained. Meanwhile, our workflow uses only one ControlNet, which is the Canny-edge ControlNet, and no further training is required.

Higher Flexibility. As our workflow uses different models for different stages, it can be adapted even when the ControlNets are not released as a module on the base model but as a full model (for example, the official Flux.1 [dev] Canny model by Black Forest Labs). Hence, the higher flexibility allows us to adapt our workflow to the Flux.1 [dev] base model within one week of the release of

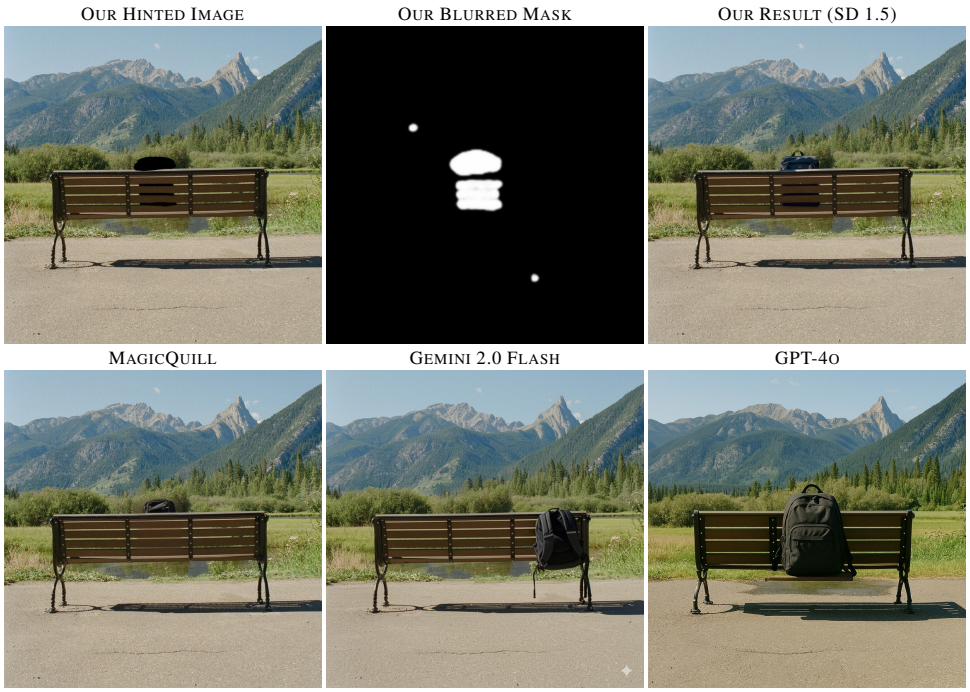


Figure 26: **Using Realistic Vision V6.0 B1, an SD-1.5-based model released in December 2023, as the backbone, our workflow outperforms both MagicQuill (with the same backbone) and latest commercial VLMs (released in March 2025).** For our workflow and MagicQuill, we used the same description prompt "black backpack on a bench". For Gemini 2.0 Flash and GPT-4o, we used the same editing prompt "add a black backpack onto this bench". Only our method results in almost perfect spatial relationships, with only a minor error at the bottom of the backpack. MagicQuill is unable to generate the lower part of the backpack, which is supposed to be visible through slits on the backrest. Gemini 2.0 Flash and GPT-4o both generate a floating backpack in front of the backrest. The original image and our better result of using Flux.1 [dev] as the backbone are shown in Figure 4.

Flux.1 [dev] Canny in November 2024, whereas MagicQuill still supports neither SDXL or FLUX models by April 2025.[26]

Better Detail Preservation. Our workflow does not downsample the color information. Hence, high frequency details are preserved from user input. In contrast, MagicQuill downsamples user color input to a 32×32 resolution during pre-processing, an information bottleneck that limits the ability of users to provide detailed color hints.

Disentangled Mask and Colors MagicQuill assumes that the user only wants to edit the region around locations where the color patch is provided. The mask is uniformly expanded in all directions from the color patch. However, consider the example of adding a backpack on a bench (Figure 26). In this case, the mask should be expanded more in the horizontal direction than the vertical direction, because the mask should not fully cover the backrest of the bench. To address this difficulty, the disentangled design in our workflow returns the freedom to the user.

Streamlined Components. First, our workflow does not uses an MLLM to guess user inputs. While MLLM provides convenience in MagicQuill for common objects, it heavily hallucinates when adding an object that is partially occluded by other objects, because the provided color patch does not have the shape of the full un-occluded object. The hallucination leads to negative user experience when challenging edits are required. Second, our workflow does not use an additional CNN to extract edges. Empirically, the Canny edges provides satisfactory performance and can be quickly extracted in our workflow.

[26] `https://github.com/ant-research/MagicQuill/issues/89`

Unlimited Resolution. By default, MagicQuill generates an image with the resolution of the whole image and replaces the masked region with generated content. This is troublesome when the user wants to iteratively perform global and local edits on the same image. Our workflow mitigates the multi-scale editing issue by proposing context dots, an intuitive and user-friendly method for specifying generation resolutions. The existing "Resolution Adjustement" functionality in MagicQuill (updated December 6, 2024) is not designed for multi-scale editing.

Multi-Purpose. By changing parameters and resolutions, our workflow smoothly transitions from an image editor to a detail enhancer, or from an inpainter to an outpainter. Examples of both transitions can be found in Figure 23. Neither detail enhancing nor outpainting is supported by MagicQuill.

Due to the above advantages, our method outperforms MagicQuill on a challenging example (Figure 26), with the same Realistic Vision V6.0 B1 (based on SD 1.5) model[27] as the backbone. To ensure fair comparison, we take the first-shot, single-step result from our method, but try our best to tune the hyperparameters and to draw accurate shapes for MagicQuill. After over 20 attempts, we are still unable to correctly add the backpack onto the bench using MagicQuill.

Here, we additionally provide comparison with GPT-4o (March 2025) and Gemini 2.0 Flash (March 2025). The fact that an SD-1.5-based model (December 2023) outperforms latest commercial VLMs suggests intrinsic limitations of text-only image editing frameworks.

We encourage readers with more experience in the baseline methods to independently verify their performance upper-bound.

O Limitations

Our workflow has the following limitations.

Texture. Our workflow performs seamless inpainting when the original image is either (1) the output of the workflow or (2) the text-to-image output of the same base model used in the workflow. If the goal is to edit a real photo or an image generated by another base model, the distribution mismatch will cause minor texture mismatches that are impossible to fix perfectly. While such artifacts are barely noticeable with human eyes, they would not escape heuristics-based editing artifact detectors.

Reference Images. Our workflow does not allow copying all details of a reference image in a single editing step. If the user requires editing complicated apparel, including larger pieces of clothing and smaller accessories, we recommend using multiple editing steps, varying the context size according to the item size, and adding one item at a time.

Local Minima. If the base model has some strong local minima, our workflow is unable to help the user generate content out of those local minima. Two examples from the base model (Flux.1 [dev] with a LoRA) we use are (1) a fixed hairstyle regardless of the edge hints and (2) two patellas on one knee regardless of the color hints. To mitigate these artifacts, a convenient and successful workaround is to increase the Canny model steps (correspondingly reducing the base model steps), since the Canny model and the base model seldom suffer from the same local minima. The resulting image after successful edits is shown in the right panel of Figure 24.

User Input. To achieve a high editing quality, a user needs to provide mask and color guidance and to tune hyperparameters. This is harder for the user than baseline methods, where a user only needs to select a better-looking image, if there exists such an image in the outputs. We have not thoroughly tested our workflow in fully automated settings.

P Workflow Implementation

The steps in our workflow can be easily implemented (or have already been separately implemented) in the most popular diffusion model Web UIs (Stable Diffusion Web UI Automatic1111, ComfyUI, and Stable Diffusion Web UI Forge). While there are myriad other methods which perform similar

[27] `https://civitai.com/models/4201?modelVersionId=245598`

functionalities, we narrow down the necessary components to the set of our choice. The novelty of our workflow is that we achieve greatly improved image editing quality with this small set of simple, user-friendly steps.

For Flux.1 [dev], our workflow is best implemented in ComfyUI. To implement our workflow in ComfyUI, we refer to the following YouTube and Reddit tutorials:

1. `https://www.youtube.com/watch?v=mI0UWm7BNtQ`
2. `https://www.youtube.com/watch?v=GvZXK4phJmE`
3. `https://www.reddit.com/r/comfyui/comments/1d1x5ek/dual_prompting_with_split_sigmas`

Other than Flux.1 [dev], we thoroughly test our workflow with BoleroMix (Pony) v1.41,[28] a Japanese animation-style base model derived from SDXL [18]. For this one and other SDXL-derived models, the user can implement the workflow by simply enabling relevant add-ons in Stable Diffusion Web UI Automatic1111. We also test our workflow in other models at a smaller scale, where all steps in our workflow similarly work.

We have tested our workflow on one NVIDIA RTX A6000, NVIDIA A100 (80 GB), or NVIDIA GeForce RTX 4090. On A6000, each editing step using Flux.1 [dev] (1024×1024 resolution and 30 diffusion steps) only takes around 30 seconds. The price of a server with a single A6000 can be as low as 0.5$ per hour,[29] equivalent to 0.004$ per editing step. This is one tenth of the cost of DALL·E 3, which is 0.04$ per image.[30] Due to the low success rate of DALL·E 3 on hard editing tasks (Appendix C.4), the cost gap is only larger in real scenarios.

The workflow, together with the installation instructions for popular Web UIs, has been released in a GitHub repository. Please see the abstract of the paper for the link.

Q Societal Impact

For the first time, our workflow unlocks the ability to perform photo-realistic edits with perfect details using diffusion models. Since the workflow is fully open source, malicious users may misuse the workflow, leading to negative societal consequences. However, as there exist various techniques for detecting AI-generated content, we do not believe the edited images will pose an immediate threat to the public. Still, we strongly encourage the users to adhere to all applicable laws and respect moral standards when generating images with our workflow and using them.

[28] `https://civitai.com/models/448716?modelVersionId=629179`

[29] `https://cloud.vast.ai/` (January 2025)

[30] `https://openai.com/api/pricing/`

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly state the task and multiple contributions of our proposed method. The contributions are thoroughly discussed in corresponding sections in the main paper and appendices.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please see Appendix O.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: We do not provide any theoretical results in the paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We release the code and our own inputs and outputs in the supplemental material. The reader should be able to fully reproduce our results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Open access is our top priority. The data and code is included in the supplemental material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We clearly specify all hyperparameters and the way they are chosen (Appendix D).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In Table 2, we report standard deviation of the metrics. Because the results are close, some comparisons are not statistically significant. Hence, we further conduct extensive qualitative study to compare the models.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: For the whole project, we used a single local NVIDIA RTX A6000, including preliminary and failed experiments. We also estimate the cost of real application scenarios in Appendix P.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work does not cause any potential harm.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Please see Appendix Q.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We require that the users adhere to usage guidelines (Appendix Q).

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets are properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We include the code in the supplemental material. The code will be open source after the review period. We will also release tutorial and documentation for ease-of-use.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: Please see Appendix E.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We only conduct a small-scale human preference study with 9 annotators. The annotation takes about 1 hour for each annotator, so there are no potential risks.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA] .

    Justification: Our proposed method does not use any LLM as a component.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.