Streamlining Image Editing with Layered Diffusion Brushes

Peyman Gholami Robert Xiao
University of British Columbia
{peymang, brx}@cs.ubc.ca

https://layered-diffusion-brushes.github.io/

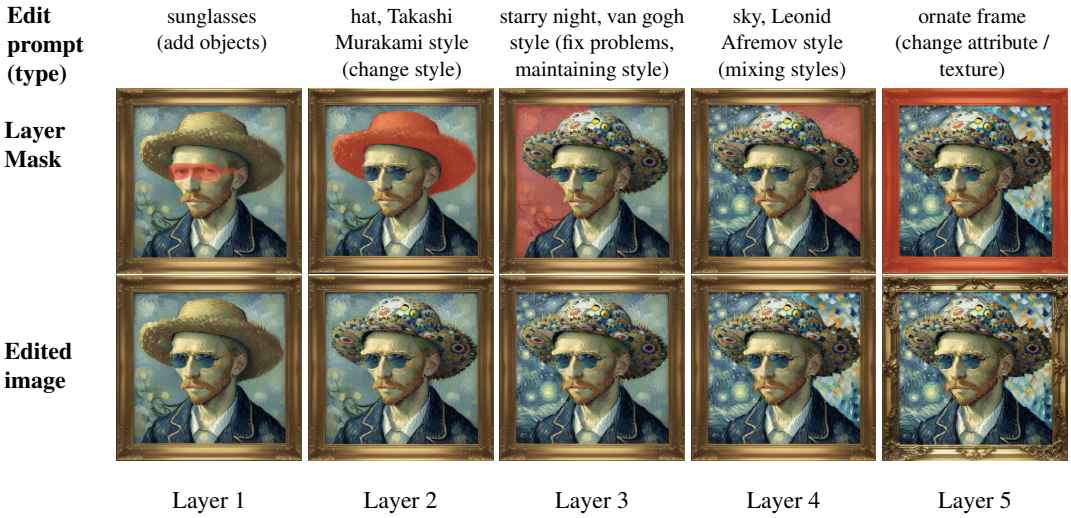


Figure 1. hierarchical image editing with Layered Diffusion Brushes: LDB is capable of creating and stacking a wide range of independent edits, including object addition, removal, or replacement, colour and style changes/combining, and object attribute modification. Each edit is performed independently, and users are able to switch between the edits seamlessly.

Abstract

Denoising diffusion models have emerged as powerful tools for image manipulation, yet interactive, localized editing workflows remain underdeveloped. We introduce Layered Diffusion Brushes (LDB), a novel framework that facilitates real-time and iterative image editing with fine-grained, region-specific control. LDB leverages a unique approach that caches intermediate latent states within the diffusion process, enabling users to apply prompt-guided edits via masks in a non-destructive, layered manner. Key innovations include latent caching for significant speed enhancements (achieving edits in under 140ms on consumer GPUs) and redefining layering for diffusion models with an order-agnostic system that allows for independent manipulation and stacking of edits, even in overlapping regions. An editor implementing LDB, incorporating familiar layer concepts, was evaluated through user study and quantitative metrics. Results demonstrate LDB's superior speed alongside comparable or improved image quality, background preservation, and edit fidelity relative to existing state-of-the-art techniques across various sequential image manipulation tasks. The findings highlight LDB's potential to significantly enhance creative workflows by providing an intuitive and efficient approach to diffusion-based image editing and its potential for expansion into related subdomains, such as video editing.

1. Introduction

Image editing has undergone transformative advancements with the rise of text-to-image (T2I) generative models, enabling unprecedented creative expression through textual guidance. These models, including Generative Adversarial Networks (GANs) [18], Variational Autoencoders (VAEs), and Denoising Diffusion Models (DMs) [21], have redefined image synthesis and manipulation. Among these, DMs [54] have emerged as the state of the art due to their training stability, high-fidelity outputs, and versatility across tasks like inpainting [35], super-resolution [51], and style transfer [20]. However, despite their capabilities, a

critical gap remains: enabling **real-time, localized, and iterative edits** that align with professional workflows, where artists demand precise control over specific regions without disrupting the global composition.

Existing DM-based editing methods face several core challenges. First, their stochastic nature often necessitates numerous generations to achieve desired results [5]. Second, they lack intuitive mechanisms for layered, non-destructive editing—a cornerstone of tools like Adobe Photoshop [26]—where edits can be independently adjusted, stacked, or removed. Third, while mask-guided approaches enable regional control, they struggle with seamless blending, artifact-free transitions, and real-time feedback. These limitations restrict their adoption in creative pipelines, where rapid iteration and granular control are critical.

To address these challenges, we propose *Layered Diffusion Brushes (LDB)*, a novel framework based on Latent Diffusion Model (LDM) [48] that integrates mask-guided diffusion with a non-destructive layered editing paradigm.

At its core, LDB introduces new noise patterns into the image latents during reverse diffusion, guided by both the user-specified mask and the edit prompt. This preserves the original context while seamlessly integrating localized edits. For streamlined adjustments, we implement an intuitive user interface (UI) with a layering system to enable consecutive edits. (Fig. 1)

Specifically, as key contributions, LDB introduces:

- **Latent Caching for Real-Time Edits:** By reusing intermediate denoising states from initial generation, edits bypass redundant computations and achieve as low as 140 ms per edit on 512×512 images (53× faster than BrushNet [29] using the same consumer GPUs).
- **Non-destructive Layered Editing:** LDB introduces an order-agnostic layering mechanism by redefining the concept of layers for DMs, enabling:
  - Region-targeted adjustments with background preservation, using mask-prompt pairs,
  - Stacking, toggling, or deleting layers without cross-interference—even in overlapping regions,
  - Post-hoc revision of edits while preserving underlying content.
- **Seed-Driven Exploration:** Our UI provides familiar "brush" and "scroll" gestures to enable instant exploration of variations by modulating noise seeds, bridging stochastic generation with deterministic refinement and instant feedback.

We validate LDB through extensive experiments and a user study with graphic designers. Quantitatively, LDB outperforms state-of-the-art methods in terms of speed and image quality and is comparable in terms of edit fidelity. The user study revealed superior usability and creativity support in iterative design. Additionally, LDB is a plug-and-play, training-free system adaptable to existing models and applications, and we demonstrate this by applying LDB to the task of video editing.

2. Related Works

2.1. DM-based image editing

Image editing is the task of modifying existing images in terms of appearance, structure, or composition, ranging from subtle adjustments to major transformations. Unlike GAN-based approaches [1, 33, 42], which are prone to limitations in inversion stability [47] and localized control [6], diffusion-based methods harness the power of controllable, high-quality DMs in various image-editing tasks, including text and image-driven image manipulation studies [13, 24, 32, 35].

Instruction-based text editing methods [9, 16, 17, 19, 63] typically train DMs on instruction-image pairs. For example, InstructPix2Pix [9] is trained using synthetic pairs from Stable Diffusion [48] and Prompt-to-Prompt [20]. However, expressing nuanced edits solely through text instructions remains challenging, particularly for object-specific style or color changes.

Mask-based methods [4, 5, 13, 62] sample within specified regions. While effective for localized edits, they can introduce unintended global changes, especially problematic in sequential editing, and may struggle with complex edits. For instance, Blended Latent Diffusion's lossy VAE latent space hinders perfect reconstruction even before noise addition [5]. Though a background reconstruction strategy is included, it increases computation and may still yield incoherent results for complex edits. Conversely, our method directly modifies the original latent space, enhancing context preservation, enabling natural blending, and improving performance.

Attention-based editing manipulates cross-attention maps to guide the image generation process toward the desired modifications [20, 43]. These methods generally face challenges in achieving fine-grained edits without unwanted global modifications. Yang et al. [60] attribute unintended changes to inaccurate attention maps and propose attention focusing. Inversion-based methods like ILVR [11], Textual Inversion [15], and DreamBooth [49] focus on context modification while preserving subjects. DDIM inversion converts images to noisy latents, and sampling generates edited results based on prompts. We employ Direct Inversion [28] for efficient real image latent inversion.

Image inpainting involves replacing or restoring the missing regions within an image while maintaining global coherency [59]. Many inpainting works [37, 48, 61, 68] require using a fine-tuned DM specifically designed for inpainting tasks, limiting their applicability. Some, including SmartBrush, which uses object-mask prediction guidance [58], offer more flexibility. PowerPaint [68] introduces

learnable task embeddings for improved control. While these models effectively generate new content, they are generally unsuitable for making small, targeted adjustments [4, 35, 50]. Inspired by ControlNet [66], BrushNet [29] builds a decomposed plug-and-play dual-branch DM, but struggles with real-time interaction due to its computational overhead. In Sec. 4.1 we compare LDB with several inpainting techniques.

2.2. Layered and Consecutive Image editing

Layer-based image editing is fundamental in computer graphics [44], and recent works integrate this concept into AI methodologies [6, 52]. Layered representations enable dynamic manipulation of image components, transforming single images into multi-layered structures.

LayeringDiff [31] decomposes images into foreground and background. ParallelEdits [25] uses attention for efficient multi-aspect text edits. MAG-Edit [38] employs a two-layer process with attention injection to a single edit from background. Joseph et al. [27] highlight error accumulation in sequential editing, where artifacts compound across edits. Collage Diffusion [52] uses layer alpha masks to modify cross-attention, generating harmonized images respecting scene composition. Their framework, based on modified Blended Latent Diffusion [5], assumes pre-layered input and synthesizes cohesive outputs. A key advantage of our method over Collage Diffusion is the ability to create and adjust fully independent layers.

2.3. Accelerated Generation using Caching

Caching and reusing intermediate features has proven effective for accelerating DM inference through reducing redundant computations. Several works have utilized caching in diffusion transformers (DiTs) for video generation. DeepCache [36] reuses high-level U-Net features in video generation, while AdaCache [30] dynamically adjusts cached residuals based on temporal content. Cache Me If You Can [56] employs block caching by reusing outputs from layer blocks of previous steps during inference. For image generation, Approximate Caching [2] reuses intermediate latents created during prior image generation processes for similar prompts. We employ a similar strategy through caching key latent representations and adapt it specifically for interactive image editing, enabling the real-time feedback that is crucial for creative workflows.

3. Method

We use an LDM-based variant of image generative models and make intermediate adjustments to the latent space using the pre-trained LDM, similar to [5, 35]. Therefore, LDB requires no additional training or fine-tuning of the underlying LDM; all modifications are applied during the reverse diffusion process.

We adopt the standard LDM formulation, where image generation begins with a sample from a Gaussian distribution, $Z_0 \sim \mathcal{N}(0, \sigma_{max}^2 I)$ and is iteratively denoised through a sequence of steps N, resulting in a series of latents Z_i corresponding to decreasing noise levels σ_i, where $\sigma_0 = \sigma_{max} > \sigma_1 > \cdots > \sigma_N \approx 0$.

As demonstrated in Fig. 2, the overall LDB pipeline comprises three key stages: **initial image generation (or inversion)**, **latent caching**, and **iterative layered editing.**

We first initialize the sample $Z_0 = \epsilon_0$ and noise level σ_0 ($i = 0$) if the image is generated using the DM. If a real image is used, we obtain the initial noise latent using inversion. We use Direct Inversion [28] due to its high speed and comparable performance compared to other inversion methods, including Null-Text Inversion [41] and Negative-Prompt Inversion [40]. The noisy sample then undergoes the diffusion process, caching certain intermediate latents to facilitate editing.

3.1. Latent Caching

To enable rapid, interactive editing with instant exploration and feedback, we employ latent caching to reuse intermediate representations in subsequent editing steps, minimizing redundant computations. We store two key intermediate latents as follows:

- **Regeneration Latent $\mathbf{Z_r}$:** At diffusion step $r = N - n$, where N is the total number of diffusion steps for initial image generation and n is the number of editing steps, we cache the latent Z_r, which serves as the starting point for all subsequent edits. By reusing Z_r, we avoid recomputing the initial denoising steps for each new edit, significantly speeding up the editing process (from N denoising steps to n). Effectively, Z_r represents a partially denoised latent state that retains the global image structure but is still malleable enough to accommodate localized edits.
- **Blending Latent $\mathbf{Z_b}$:** We cache the latent at diffusion step b which is specifically used for the layer merging process (Algorithm 1, line 7). We set $b = N - 2$ for maximum background preservation (as discussed in Sec. 4.3). Z_b represents a more denoised latent compared to Z_r, capturing more refined image details while still allowing for seamless blending of new edits into the existing image context. Utilizing this cached blending latent ensures smoother integration of edits and reduces visual artifacts at layer boundaries during the merging process.

3.2. Layered Diffusion Brushes Editing

To initiate an edit, the algorithm begins by generating a new noise pattern $Z_0' = \epsilon_k'$, sampled from $\mathcal{N}(0, \sigma^2 I)$ using a different seed S', and scaling it to match the variance of the cached latent Z_r. This ensures that the additive noise stays in a reasonable range from the latent for editing, preventing visual artifacts. Z_0' is then added to the regeneration latent

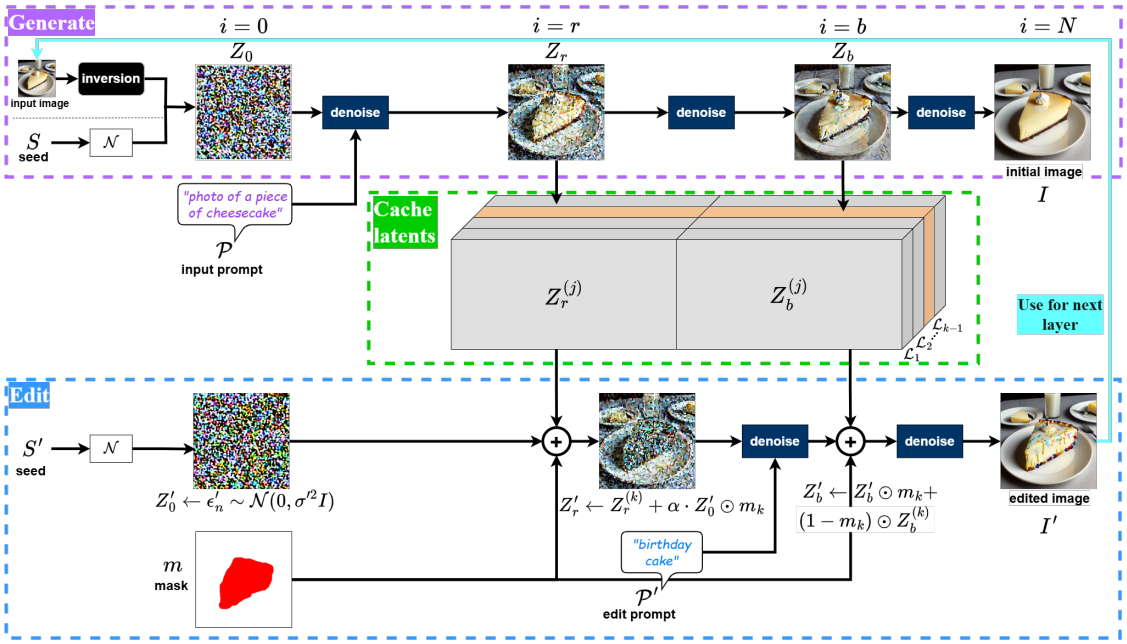


Figure 2. Overview of the Proposed Method: The top box shows standard DM-based image generation from noisy latent Z_0 and prompt $\mathcal{P}$. The middle section depicts the latent caching module, storing and retrieving intermediate latents for different layers. The bottom box illustrates the editing process: a new seed S' merges with the original latent at step r using mask m and strength control α. Diffusion continues until step b, where modified and cached latents blend to generate the final edited image.

Z_r, controlled by the mask m and strength α.

In the editing stage, at step b, a new noisy sample is merged with the cached blending latent using the strength control and the mask, resulting in Z_b'. Subsequently, the new latent is progressively denoised from steps b through N and processed through the VAE to output edited image I'. Algorithm 1 presents the pseudocode for the editing process for a single layer (for simplicity):

Algorithm 1: LDB editing process (single layer)

Input : Edit prompt $\mathcal{P}'$, Mask $m \in [0,1]^{H \times W}$, Random seed S', Strength α, Number of edit steps n, Regeneration latent Z_r, Blending latent Z_b

Output: Edited latent Z_N'

$Z_0' \leftarrow \epsilon_{n_k}' \sim \mathcal{N}(0, \sigma^2 I)$ // sampled using seed S'

$Z_0' \leftarrow \sqrt{Var(Z_r)} \cdot Z_0'$ // scale new sample

$Z_0' \leftarrow Z_r + \alpha \cdot (Z_0' \odot m)$ // noise injection

for $i = 0, 1, \ldots, n$ **do**

  $Z_{i+1}' \leftarrow DM(Z_i', \mathcal{P}', i, S')$

  if $i == b$ **then**

    $Z_b' \leftarrow Z_b' \odot m + Z_b \odot (1 - m)$ // blending

end

Return Z_N'

3.3. Layer Formulation

Unlike prior works that rely on transparent decomposable layers [65] or explicit object segmentation [52], we redefine a layer as a self-contained set of reproducible parameters that govern localized edits. For layer $\mathcal{L}_k$, we formalize this as a generalized version of parameters in Algorithm 1:

$$\mathcal{L}^{(k)} = \left(\mathbf{S}'^{(k)}, \mathbf{m}^{(k)}, \mathbf{v}^{(k)}, \mathbf{Z}_r^{(k)}, \mathbf{Z}_b^{(k)}, \alpha^{(k)}, n^{(k)}, \mathcal{P}'^{(k)}, j\right) \quad (1)$$

- $\mathbf{S}'^{(k)} \in \mathbb{Z}^+$: Seed space for stochastic variations
- $\mathbf{m}^{(k)} \in [0,1]^{H \times W}$: Edit mask
- $\mathbf{v}^{(k)} \in \{0,1\}$: Visibility state
- $\mathbf{Z}_r^{(k)}, \mathbf{Z}_b^{(k)} \in \mathbb{R}^{C \times H \times W}$: Regeneration/blending latents
- $\alpha^{(k)} \in [0,1]$: Layer strength value
- $\mathbf{n}^{(k)} \in [0, N]$ Number of denoising steps
- $\mathcal{P}'^{(k)}$: Edit prompt
- $j \in \mathbb{Z}^+$: Index of last layer index.

Notably, within a given layer $\mathcal{L}_k$ with previous layer of $\mathcal{L}_j$, cached latents $\mathbf{Z}_r^{(j)}$ and $\mathbf{Z}_b^{(j)}$ inherently incorporate the cumulative edits from all preceding layers. This is because when an edit is applied to layer $\mathcal{L}_k$, the input to the diffusion process is derived from the *already edited* output of layer $\mathcal{L}_j$ and the algorithm always keeps the last layer updated. Therefore, any modification in a previous layer automatically propagates through the subsequent layer calculations. By defining Φ as a single-layer latent generation and

caching step as:

$$(Z_r^{(k)}, Z_b^{(k)}) = \Phi(\mathcal{L}^{(k)}, \mathcal{L}^{(j)}) \quad (2)$$

in essence, if a given layer $\mathcal{L}^{(i)}$ (where $i < k$) is removed or its visibility $\mathbf{v}^{(i)}$ is toggled, the operator Φ will be recursively invoked to recreate all latents for layers from $\mathcal{L}^{(i)}$ to $\mathcal{L}^{(k)}$. This recomputation, accelerated by latent caching, is automatically triggered and typically completes within milliseconds to a few seconds, depending on the number of layers. This design allows edits to remain independent yet seamlessly integrated into the final composition.

3.3.1. Overlapping Regions

A key advantage of layered editing in LDB is the ability to create overlapping edits, where modifications in one layer can partially or fully affect regions edited by previous layers. This requires careful handling of the regeneration latent, Z_r, for each layer to ensure that changes in visibility or content of higher layers are accurately reflected in subsequent layers, even in overlapping regions.

By default, all layers use the initial image's latent (Z_r) as their regeneration latent. However, this approach fails to account for overlapping edits from preceding layers. To address this, when processing a layer k, we compute its regeneration latent by inverting the output image of the previous layer ($I'^{(k)}$) as shown using the feedback arrow on Fig. 2. This inversion yields $Z_0^{(k)}$, which is then sent through the generation stage in LDB. Both $Z_r^{(k)}$ and $Z_b^{(k)}$ are cached for efficient processing (as shown in Fig. 3).

This mechanism enables precise control and seamless integration of edits across overlapping regions. Changes to any layer propagate correctly without introducing artifacts, offering flexibility and fine-grained control.

3.4. User-Interface and Interaction Design

To develop a practical tool for artists and designers, we designed an custom UI that balances control and simplicity. The UI allows users to generate, upload, and edit images, manage layers, and adjust parameters seamlessly. Two interaction modes streamline edits (Fig. 4):

Box Mode: Users can click or drag on the image to move a resizable square mask around. This option enables a quick and interactive exploration of how various parts of the image will change in response to a given set of editing settings (prompt and strength), simply by moving the cursor.

Custom Mask Mode: Users can draw free-form masks over the desired around and navigate between new generation samples by scrolling the mouse up or down while hovering over the image, allowing them to rapidly explore variations on their edit.

We propose a workflow where users first position edits spatially using Box Mode, then refine mask geometry and appearance details via Custom Mask Mode.

Layering capabilities include stacking, visibility toggling, and deletion. Each layer is independently modifiable. Detailed information on the UI design user interactions and a demo video can be found in supplementary materials [55].

4. Experiments

4.1. User Study

We conducted a user study in order to evaluate the effectiveness of LDB for providing targeted image fine-tuning, using two other well-known existing image editing tools, InstructPix2Pix (IP2P) [9] and Stable Diffusion Inpainting (SDI) [48] as baselines for comparison.

We recruited a cohort of seven expert participants with extensive experience in using image editing software. As part of our participant selection criteria, we ensured that all participants possessed at least a basic level of familiarity with AI image generation techniques [3, 39] and were regular users of editing software, such as Adobe Photoshop [26] for creating visual art.

4.1.1. Study Procedure and Task Description

Each user engaged in two types of tasks: free-form tasks where users generated an image for editing using a fixed prompt and seed (type 1), and pre-determined tasks where the user worked with existing real images from the MagicBrush dataset [64] (type 2).

For the type 1 tasks, we selected specific types of edits that showcase various functionalities and capabilities of the system, including:

1. Stack layers and create sequential edits (draw with LDB)
2. Modify attributes and features of objects
3. Correct image imperfections and errors
4. Enhance discernibility of similar objects
5. Target specific regions for style transfer, refine aesthetics

Type 2 tasks were more structured, with the mask, edit prompt, and input images provided by the dataset. The dataset provides manually annotated masks and instructions for each edit. We selected a subset of 35 input images, each containing up to three layers of edits. Users refined masks/parameters if necessary and completed editing tasks.

Figure 5 presents examples of edited images for each technique on the MagicBrush dataset. As demonstrated, the LDB method is able to effectively make targeted adjustments which are well-integrated with the images.

4.1.2. Evaluation Survey results

The participants completed a three stage evaluation survey following the image editing tasks. The first part included a System Usability Scale (**SUS**) form to rate the usability, ease of use, design, and performance of each method. SUS is a standard usability evaluation survey widely used in user-experience literature [8]. Overall, participants indicated that they are more likely to use LDB compared to

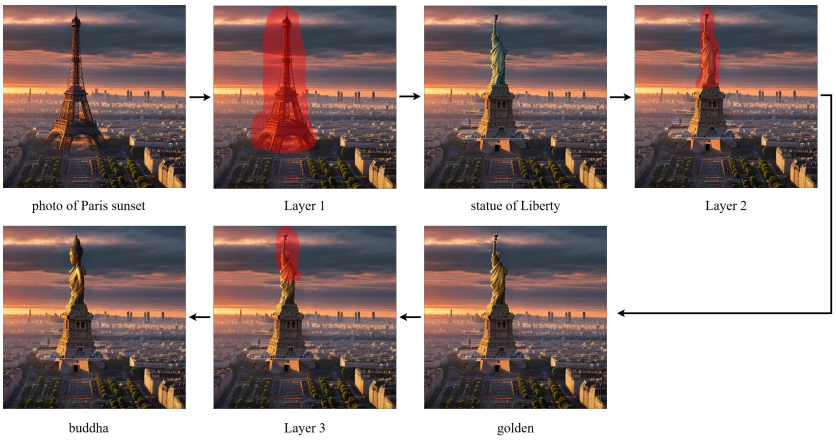


Figure 3. Overlapping edit regions in LDB: overlapping edits allow creating complex modifications that can interact with each other for more nuanced and sophisticated results. For example, one layer can modify the color of an object, while another overlapping layer can change its shape, with the final result being a combination of both edits.

(a) Box option with moving cursor (b) Custom mask option with mouse scroll

Figure 4. Box and Custom Mask Options: In box mode, users click the target region's center to generate edits within the specified area and can drag the box to explore variations instantly. In custom mask mode, users draw a mask over the desired region and adjust the seed using the mouse wheel or scrolling gestures to generate new variations.

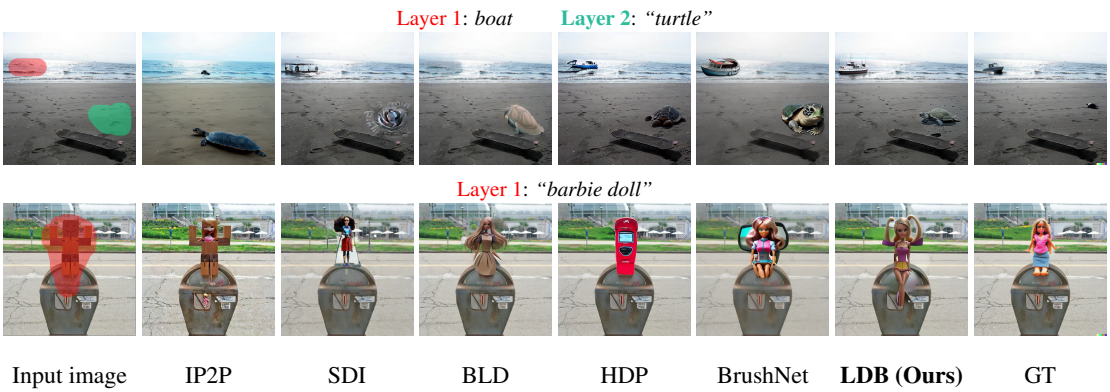


Figure 5. Editing results on MagicBrush dataset images using different methods generated by user study participants: The last column corresponds to MagicBrush's ground truth images. Edit prompts are presented on top of each row. More examples available in [55].

IP2P and SDI, and that they find it the easiest tool to use. **LDB** obtained a SUS score of **80.35%**, while **IP2P** and **SDI** achieved a SUS of **38.21%** and **37.5%** respectively.

The SUS survey was followed by a Creativity Support Index [10] survey to evaluate the system's degree of creative work support. Participants expressed positivity towards LDB, indicating that it enhanced their enjoyment, exploration, expressiveness, and immersion, while also deeming the results worth their effort. Lastly, the survey was followed by a semi-structured interview where participants appreciated the intuitiveness, ease of use and versatility of LDB. Further details about the study, interview, results, and discussion can be found in supplemental materials [55].

4.2. Quantitative Analysis

To quantitatively evaluate the performance of LDB, we employed a comprehensive suite of metrics, aligning with established practices in image editing evaluation.

Specifically, for text-image alignment, we utilized CLIP Score (**CS**) [45], calculated on the entire edited image using a global descriptor, and local CLIP Score (**CS-L**), calculated on the masked regions. We adopted Learned Perceptual Image Patch Similarity (**LPIPS**) [67] and Peak Signal-to-Noise Ratio (**PSNR**) [22] for evaluating content preservation and pixel-level fidelity in unmasked regions. Furthermore, to gauge overall image quality and aesthetic appeal, we incorporated Aesthetic Score (**AS**) [53], Image Reward (**IR**), and Human Preference Score v2 (**HPS**) [57], the latter two reflecting human-aligned preferences.

We compared **LDB** against a diverse set of state-of-the-art image editing and inpainting methods, including InstructPix2Pix (**IP2P**) [9], Stable Diffusion Inpainting (**SDI**) [48], HD-Painter (**HDP**) [37], **BrushNet** [29], and Blended Latent Diffusion (**BLD**) [5].

Quantitative results are summarized in Tab. 1, obtained from the MagicBrush selected dataset. For fair compari-

Method	Image Quality			Masked Region Preservation		Text Alignment			Time (s)
	IR $_{\times 10}$ ↑	HPS $_{\times 10^2}$ ↑	AS ↑	PSNR ↑	LPIPS $_{\times 10^2}$ ↓	CS ↑	CS-L ↑	CS-D $_{\times 10^2}$ ↑	(per edit) ↓
IP2P	-6.28	21.16	5.29	7.28	15.07	29.39	22.01	6.64	1.72
SDI	-3.92	20.88	5.48	12.20	8.70	30.08	22.15	4.11	1.84
HDP	-2.07	**23.27**	5.44	12.05	**6.13**	31.01	22.06	9.89	12.85
BrushNet	**-0.04**	22.57	**5.73**	11.55	8.75	**31.16**	**22.17**	**12.92**	7.49
BLD	-2.41	**22.80**	5.48	**12.64**	**6.94**	30.64	21.99	**10.05**	**1.41**
LDB (ours)	**0.77**	22.65	**5.74**	**12.85**	7.05	**31.04**	**22.07**	9.54	**0.26**
Ground Truth	-0.19	22.62	5.36	17.64	2.30	30.75	22.14	9.78	NA

Table 1. Comparison of methods across multiple evaluation metrics. Higher (↑) or lower (↓) values indicate better performance for each metric. We report the average scores across all layers and test images. Metrics are grouped into Image Quality, Masked Region Preservation, and Text Alignment. The **best** values are colored **green**, while the **second-best** values are colored **orange**.

son, LDB, IP2P, and SDI were evaluated on the user-edited images from our user study, while for other methods we used their default editing procedures. Inference times represent average per-edit duration measured on an NVIDIA RTX 4090 GPU with $N = 25$ diffusion steps for baseline methods and $n = 8$ steps for LDB.

4.3. Ablation Study

We perform three ablation studies for two main components of the LDB caching mechanism, *i.e.* the caching timesteps for the regeneration latent (r), and the blending latent (b). We also ablate and discuss the effect of strength control α and its relationship with n in [55].

4.3.1. Ablation on Regeneration Latent Step

The timestep r for caching the regeneration latent is a crucial parameter, as it directly influences the extent of modifications allowed during the regeneration process. To investigate its effect, we perform an ablation study by varying r while keeping the total diffusion steps N constant. This variation in r implicitly changes the number of regeneration steps (n) and necessitates adjustments to the strength parameter accordingly. Fig. 6 qualitatively illustrates the impact of different r values. As depicted, excessively small r values lead to incoherent edits and noticeable artifacts due to insufficient blending with the original image. Conversely, large r values limit the model's ability to modify the masked region, resulting in minimal changes and preserving the original content.

Quantitatively, we observe that smaller r steps (*e.g.* $r = 2$) yield higher LPIPS (0.04) and low PSNR (27.03), indicating poor image quality and fidelity. Edit fidelity scores such as CS-L also confirm that larger r steps result in lower scores (22.98), suggesting ineffective edits within the masked region. The HPS index demonstrates a higher score for mid-range steps (0.33, $r = 12$) compared to both ends of the spectrum (0.29, $r = 23$), highlighting a performance sweet spot for intermediate r values. Detailed metric graphs are available in the supplemental materials [55].

4.3.2. Ablation on Blending Latent Step

The blending latent step, controlled by the parameter b, determines when the cached regeneration latent is blended back into the diffusion process and is crucial for seamless integration of the edited region with the original image and preserving background. We conduct an ablation study by varying b while keeping r and N fixed. Fig. 7 qualitatively demonstrates the effect of different b values.

When b is small, the blending process starts prematurely, causing the edit to bleed into the background and distorting the original image context. Conversely, larger b values, representing late blending, effectively preserve the background integrity while still allowing for meaningful edits within the masked region. With larger b, the regeneration latent is diffused for a longer duration before blending, concentrating the edit within the intended area and minimizing background interference.

Quantitatively, we observed that smaller b steps ($b = n$) lead to higher LPIPS (0.17), lower PSNR (11.61) indicating a degradation in unmasked area preservation. Edit fidelity scores (CS-L) within the masked area remained stable across the spectrum while CS-D gained a much higher score for larger b values (0.32, $b = N - 1$), indicating better edit effectiveness. Our findings suggest that larger b values are generally preferable, prioritizing background preservation with effective localized editing and we choose $b = N - 2$ in the LDB algorithm. Further quantitative details and metric plots are available in the supplemental materials [55].

5. Discussion

Our experiments demonstrate that LDB establishes new benchmarks for speed and workflow adaptability in diffusion-based image editing. Key findings include:

Enhanced Control via Layering: LDB's layer formulation provides non-destructive editing, enabling iterative refinement and complex compositions. Participants high-

Figure 6. Ablation on regeneration latent step r (increasing left to right). Small r results in strong prompt adherence ("cat") but introduces artifacts. Large r (near N) leads to insufficient modification, retaining the original "dog". An intermediate r achieves the best balance of edit fidelity and background preservation.

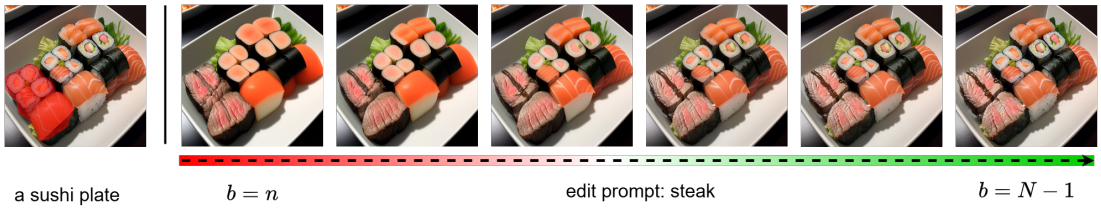


Figure 7. Ablation study on blending latent step b (increasing left to right). The prompt "steak" is applied to a sushi plate image while increasing b from left to right. At $b = n$ (left), the edit disrupts the original structure, affecting unmasked regions. As b approaches N (right), background preservation improves, and edits blend more seamlessly.

lighted how this mirrors professional-grade editing tools like Photoshop [26].

Speed and Efficiency: LDB achieves remarkable speed, 53× faster than BrushNet (evaluated on the same hardware), crucial for interactive editing. We observe that reducing diffusion steps to as few as $n = 4$ maintains reasonable quality (HPS: 0.34, CS-D: 0.35), yielding a latency of **140ms** per edit. User studies confirm *instant feedback* as a key advantage, enabling rapid iteration (tens of variations per minute *vs*. 1-2 for baselines). This speed results from efficient latent caching (Sec. 3.1), minimizing computation and memory overhead (∼1.25 MB for 10 layers).

Superior Quantitative Performance: LDB demonstrates superior background preservation (PSNR: 12.85 vs. IP2P's 7.28), maintains high edit fidelity (CS-L: 22.07), and robustly handles overlapping regions (LPIPS: 7.05, see Fig. 3), outperforming baselines. As presented in Tab. 1, LDB leads in the highest human-preference-derived metric (IR: 0.77), aligning with the user study usability (SUS: 80.35). While BrushNet slightly surpasses in CS-D, its slow edit speed limits practicality.

5.1. Limitations and Future Work

Currently, brush strength (α) and diffusion step count (n) coupling (Fig. 11) requires minor user tuning across scenarios. Although preset profiles partially address this, future work could explore adaptive parameter tuning mechanisms to further improve usability. Moreover, semantically implausible edits (*e.g*. placing a boat in the sky) remain challenging due to inherent biases within diffusion models. Integrating techniques like semantic guidance could expand plausible edit ranges. Finally, responsible deployment necessitates robust watermarking [14] and provenance tracking to mitigate misuse and ensure transparency.

5.2. Broader Applications

LDB's localized editing capability enables easy, training-free integration into diverse diffusion models and applications requiring rapid editing.

Traditional diffusion-based video editing typically propagates edits from the first frame using additional supervision (*e.g*. optical flow [34]), risking temporal inconsistencies. LDB's high fidelity background preservation and efficiency naturally address these issues.

We demonstrate preliminary success integrating LDB with Stable Video Diffusion (SVD) [7], editing the first frame and applying LDB's latent caching across frames for fast consecutive edits (see supplemental material Fig. 14). This approach opens avenues for accelerated video manipulation, 3D asset editing, and collaborative design platforms.

5.3. Conclusion

LDB reimagines diffusion-based editing through latent caching and non-destructive layering, achieving unmatched speed and control. Quantitative results and user study show superior performance in image preference, edit fidelity, time, and usability. By bridging interactive editing with high-fidelity generative models, LDB can empower artists to iterate fluidly while maintaining artistic intent.

References

[1] Rameen Abdal, Peihao Zhu, Niloy J Mitra, and Peter Wonka. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. *ACM Transactions on Graphics (ToG)*, 40(3):1–21, 2021. 2

[2] Shubham Agarwal, Subrata Mitra, Sarthak Chakraborty, Srikrishna Karanam, Koyel Mukherjee, and Shiv Kumar Saini. Approximate caching for efficiently serving {Text-to-Image} diffusion models. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1173–1189, 2024. 3

[3] Ideogram AI. Ideogram: Text-to-image generation platform. https://ideogram.ai, 2025. 5

[4] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18208–18218, 2022. 2, 3

[5] Omri Avrahami, Ohad Fried, and Dani Lischinski. Blended latent diffusion. *ACM Transactions on Graphics (TOG)*, 42 (4):1–11, 2023. 2, 3, 6

[6] Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. Text2live: Text-driven layered image and video editing. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XV*, pages 707–723. Springer, 2022. 2, 3

[7] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 8, 2

[8] John Brooke. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, 189(194), 1996. 5, 7

[9] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. *arXiv preprint arXiv:2211.09800*, 2022. 2, 5, 6

[10] Erin Cherry and Celine Latulipe. Quantifying the creativity support of digital tools through the creativity support index. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 21(4):1–25, 2014. 6, 8

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

[12] Civitai. Dreamshaper - 7 — stable diffusion checkpoint, 2024. Accessed on Feb 19, 2024. 5

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

[14] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22466–22477, 2023. 8

[15] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022. 2

[16] Zigang Geng, Binxin Yang, Tiankai Hang, Chen Li, Shuyang Gu, Ting Zhang, Jianmin Bao, Zheng Zhang, Han Hu, Dong Chen, et al. Instructdiffusion: A generalist modeling interface for vision tasks. *arXiv preprint arXiv:2309.03895*, 2023. 2

[17] Zigang Geng, Binxin Yang, Tiankai Hang, Chen Li, Shuyang Gu, Ting Zhang, Jianmin Bao, Zheng Zhang, Houqiang Li, Han Hu, et al. Instructdiffusion: A generalist modeling interface for vision tasks. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 12709–12720, 2024. 2

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 1

[19] Qin Guo and Tianwei Lin. Focus on your instruction: Fine-grained and multi-instruction image editing by attention modulation. *arXiv preprint arXiv:2312.10113*, 2023. 2

[20] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022. 1, 2

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1

[22] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010. 6

[23] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 1

[24] Minghui Hu, Yujie Wang, Tat-Jen Cham, Jianfei Yang, and Ponnuthurai N Suganthan. Global context with discrete diffusion in vector quantised modelling for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11502–11511, 2022. 2

[25] Mingzhen Huang, Jialing Cai, Shan Jia, Vishnu Suresh Lokhande, and Siwei Lyu. Paralleledits: Efficient multi-aspect text-driven image editing with attention grouping. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 3

[26] Adobe Inc. Adobe photoshop (2024 version). https://www.adobe.com/products/photoshop.html, 2024. 2, 5, 8

[27] K. J. Joseph, Prateksha Udhayanan, Tripti Shukla, Aishwarya Agarwal, Srikrishna Karanam, Koustava Goswami, and Balaji Vasan Srinivasan. Iterative multi-granular image editing using diffusion models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 8107–8116, 2024. 3

[28] Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Direct inversion: Boosting diffusion-based edit-

ing with 3 lines of code. *arXiv preprint arXiv:2310.01506*, 2023. 2, 3

[29] Xuan Ju, Xian Liu, Xintao Wang, Yuxuan Bian, Ying Shan, and Qiang Xu. Brushnet: A plug-and-play image inpainting model with decomposed dual-branch diffusion. In *European Conference on Computer Vision*, pages 150–168. Springer, 2024. 2, 3, 6

[30] Kumara Kahatapitiya, Haozhe Liu, Sen He, Ding Liu, Menglin Jia, Chenyang Zhang, Michael S Ryoo, and Tian Xie. Adaptive caching for faster video generation with diffusion transformers. *arXiv preprint arXiv:2411.02397*, 2024. 3

[31] Kyoungkook Kang, Gyujin Sim, Geonung Kim, Donguk Kim, Seungho Nam, and Sunghyun Cho. Layeringdiff: Layered image synthesis via generation, then disassembly with generative knowledge. *arXiv preprint arXiv:2501.01197*, 2025. 3

[32] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2426–2435, 2022. 2

[33] Oran Lang, Yossi Gandelsman, Michal Yarom, Yoav Wald, Gal Elidan, Avinatan Hassidim, William T Freeman, Phillip Isola, Amir Globerson, Michal Irani, et al. Explaining in style: Training a gan to explain a classifier in stylespace. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 693–702, 2021. 2

[34] Feng Liang, Bichen Wu, Jialiang Wang, Licheng Yu, Kunpeng Li, Yinan Zhao, Ishan Misra, Jia-Bin Huang, Peizhao Zhang, Peter Vajda, et al. Flowvid: Taming imperfect optical flows for consistent video-to-video synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8207–8216, 2024. 8

[35] 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. 1, 2, 3

[36] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15762–15772, 2024. 3

[37] Hayk Manukyan, Andranik Sargsyan, Barsegh Atanyan, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Hd-painter: High-resolution and prompt-faithful text-guided image inpainting with diffusion models. *arXiv preprint arXiv:2312.14091*, 2023. 2, 6

[38] Qi Mao, Lan Chen, Yuchao Gu, Zhen Fang, and Mike Zheng Shou. Mag-edit: Localized image editing in complex scenarios via mask-based attention-adjusted guidance. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 6842–6850, 2024. 3

[39] Inc. Midjourney. Midjourney: Ai image generation. https://www.midjourney.com, 2025. 5

[40] Daiki Miyake, Akihiro Iohara, Yu Saito, and Toshiyuki Tanaka. Negative-prompt inversion: Fast image inversion for editing with text-guided diffusion models. *arXiv preprint arXiv:2305.16807*, 2023. 3

[41] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6038–6047, 2023. 3

[42] Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023. 2

[43] Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. In *ACM SIGGRAPH 2023 Conference Proceedings*, New York, NY, USA, 2023. Association for Computing Machinery. 2

[44] Thomas Porter and Tom Duff. Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 253–259, 1984. 3

[45] 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. 6

[46] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1 (2):3, 2022. 5

[47] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2287–2296, 2021. 2

[48] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 2, 5, 6

[49] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation, 2023. 2

[50] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022. 3

[51] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1

[52] Vishnu Sarukkai, Linden Li, Arden Ma, Christopher Ré, and Kayvon Fatahalian. Collage diffusion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4208–4217, 2024. 3, 4

[53] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022. 6

[54] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 1

[55] Anonymous ICCV submission. Streamlining image editing with layered diffusion brushes, 2025. ID 2610. Supplied as supplemental material `2610.pdf`. 5, 6, 7

[56] Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, et al. Cache me if you can: Accelerating diffusion models through block caching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6211–6220, 2024. 3

[57] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023. 6

[58] Shaoan Xie, Zhifei Zhang, Zhe Lin, Tobias Hinz, and Kun Zhang. Smartbrush: Text and shape guided object inpainting with diffusion model. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22428–22437, 2023. 2

[59] Zishan Xu, Xiaofeng Zhang, Wei Chen, Minda Yao, Jueting Liu, Tingting Xu, and Zehua Wang. A review of image inpainting methods based on deep learning. *Applied Sciences*, 13(20):11189, 2023. 2

[60] Fei Yang, Shiqi Yang, Muhammad Atif Butt, Joost van de Weijer, et al. Dynamic prompt learning: Addressing cross-attention leakage for text-based image editing. *Advances in Neural Information Processing Systems*, 36:26291–26303, 2023. 2

[61] Shiyuan Yang, Xiaodong Chen, and Jing Liao. Uni-paint: A unified framework for multimodal image inpainting with pretrained diffusion model. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 3190–3199, 2023. 2

[62] Tao Yu, Runseng Feng, Ruoyu Feng, Jinming Liu, Xin Jin, Wenjun Zeng, and Zhibo Chen. Inpaint anything: Segment anything meets image inpainting. *arXiv preprint arXiv:2304.06790*, 2023. 2

[63] Ziyun Zeng, Hang Hua, Jianlong Fu, Jiebo Luo, et al. Promptfix: You prompt and we fix the photo. *Advances in Neural Information Processing Systems*, 37:40000–40031, 2025. 2

[64] 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. 5

[65] Lvmin Zhang and Maneesh Agrawala. Transparent image layer diffusion using latent transparency. *arXiv preprint arXiv:2402.17113*, 2024. 4

[66] 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. 3

[67] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6

[68] Junhao Zhuang, Yanhong Zeng, Wenran Liu, Chun Yuan, and Kai Chen. A task is worth one word: Learning with task prompts for high-quality versatile image inpainting. In *European Conference on Computer Vision*, pages 195–211. Springer, 2024. 2

Streamlining Image Editing with Layered Diffusion Brushes

Supplementary Material

6. UI and interaction design

Fig. 8 provides an overview of the user interface. As demonstrated, the UI comprises the following primary sections (each section highlighted with the corresponding number on the image):

1. Model Section
   - This section in the UI enables users to load various model combinations, including pre-trained models, schedulers, and LoRA [23].
2. Generation section
   - This section allows users to either generate a new image using a seed and prompt combination, or upload a real image that will be inverted.
3. Image Canvas
   - This canvas serves as the workspace where users interact with and make edits to images.
4. Editing Section
   - This section provides controls to create and modify different layers to make the desired edits.

We provide the ability to stack and hide/unhide layers, similar to traditional image-editing tools.

When editing a layer, we provide the choice of box mode or brush mode. In box mode, the mask is a square shape controlled by the "brush size" parameter. As the box is dragged around the image, the seed value will automatically increment, providing a continuous stream of new edits. The user may stop dragging when a suitable edit is seen.

In brush mode, the mask is an arbitrary shape that can be added to or subtracted from using a circular brush tool. The size of the brush is controlled by the "brush size" parameter. In this mode, the user can scroll a mouse wheel or use a scrolling gesture to increment or decrement the seed, allowing them to rapidly explore the space of potential edits and return to any edit that appears suitable.

6.1. Hyperparameters

To provide a balance between usability and complexity, we provide control over a number of hyperparameters: number of regeneration steps, "brush strength", brush size and seed number. Each hyperparameter is designed to be largely orthogonal to the other parameters, enabling them to independently affect the appearance of the edit without the need to simultaneously adjust multiple inputs.

- **Number of regeneration steps** (n): An integer value that specifies the number of steps that the LDB will run to make the edit. Changing n effectively changes the strength of the modification as well as the processing time.
- **Brush Strength** (α): A number that indirectly controls the α value in (Eq. (3)) which controls how strong the initial noise pattern should be. The user-specified alpha, α^*, has a value between 0 and 100, which will be scaled using the following equation:

$$\alpha = \frac{\sqrt{\left|\frac{\alpha^*}{100} \cdot \left(\sigma - 2 \cdot \frac{\mathrm{Cov}(Z_r^{(k)}, Z_0')}{\mathrm{Var}(Z_r^{(k)})}\right)\right|}}{\sqrt{\frac{\sum_{i=1}^{W}\sum_{j=1}^{H}[m_{ij}\neq 0]}{W}}} \tag{3}$$

  where Z_0' and $Z_r^{(k)}$ are the new noise latent and latent for regeneration respectively (as noted in Algorithm 1), σ is the acceptable range for the variance of the $Z_r^{(k)}$ (we used σ=0.25), m is the corresponding mask, and W is the width of Z_{n_k} (W=512).
  This formula is designed to ensure that any fixed value of the user-provided α^* value produces similar effects on the image even as the number of regeneration steps or the brush size/mask size are changed, thus making it more logically independent from the other parameters.
- **Seed Number** (s'): An integer number that will be used for generating the Gaussian noise pattern in the specified region. As with normal image generation, the UI provides buttons to randomize the seed or reuse the previous seed. Moving the box around (in box mode) or using the scroll wheel (in custom mask mode) will adjust the seed automatically.
- **Brush Size** d: An integer value that dictates the radius of the box when utilized in box mode, or the size of the brush in custom mask mode (in pixels).

7. Additional Qualitative Examples

Fig. 9 and Fig. 10 present examples of Type 1 tasks (freeform) and Type 2 tasks (MagicBrush) respectively. All the images were edited by participants during the user study.

8. Ablation Study Details

8.1. Ablation on Mask strength control

The magnitude of the edit applied by LDB is jointly governed by the number of edit steps (n) and the mask strength control (α). These parameters control the amount of intermediate noise added to the latent image. Fig. 11 illustrates the effect of varying α. As shown, excessively high α values (right), representing strong edits, prevent the LDM from effectively denoising, leading to artifacts. Conversely, insufficient α results in negligible edits. Furthermore, n and

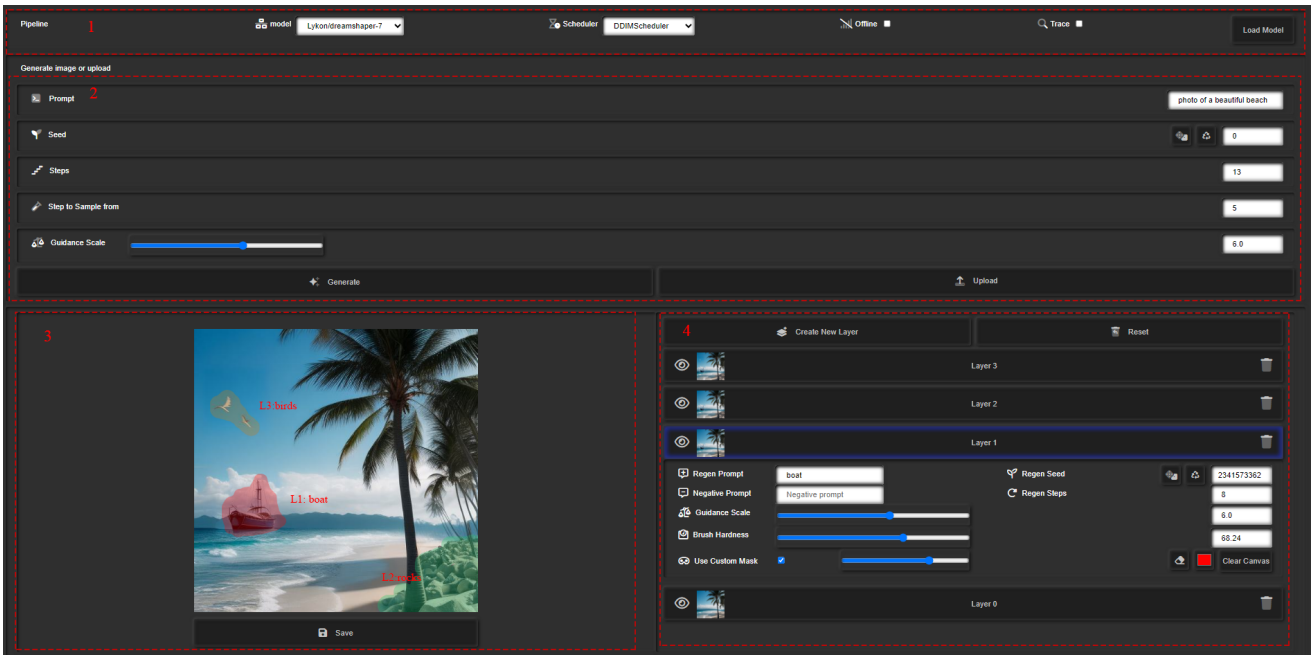


Figure 8. Design of the LDB's UI: The name and functionality of each section are described in the text. In this example, the user has created three layers, visualized on the image canvas, along with the mask and edit prompt. The selected layer in this picture is Layer 1.

α exhibit a coupled relationship. When noise is introduced later in the diffusion process (higher n), the model has less denoising capacity, necessitating a higher α to achieve a noticeable edit. Conversely, with earlier noise injection (lower n), a sufficiently large α is required to prevent the additive noise from being entirely diffused away in the initial denoising steps. Therefore, optimal editing requires careful consideration of both n and α, with α needing adjustment based on the chosen n to balance edit strength and image quality. In our UI, we formulate the translation from α^* to α to decouple these two parameters by factoring in the variance and covariance of the intermediate latent, thus automatically adjusting α when n changes.

8.2. Caching Latents Ablation Metrics

Fig. 12 and Fig. 13 present the graphs for quantitative metrics on the ablation studies as discussed in Sec. 4.3.

9. Video Editing Examples

We integrated LDB with several diffusion image transformers (DiT) and spatio-temporal video generation models. In Fig. 14, we demonstrate examples of video editing by integrating LDB into SVD [7].

10. User Study Details

10.1. Procedure and Task Description

The user cohort comprised four females and three males, with an average age of 30.4 years. Two participants were proficient in image generative models and Stable Diffusion, while the remaining five were graphic design students who used Adobe Photoshop and Illustrator on a daily basis. The study was conducted remotely; participants were provided a link to access the tool.

The study started with a brief introduction to each of the methods. Following this, participants received a short tutorial on how to navigate the user interface (UI). Subsequently, they were provided with a 5-minute window to explore the various options and sections of the tool, becoming familiar with the use of each section.

A dedicated task section was incorporated into the user interface (UI) specifically for the user study. Each type of task comprised three rounds of edits using the three methods: LDB, IP2P, and SDI.

Each user was assigned a unique user ID, and tasks were randomly selected and pre-assigned to users. Throughout the study, users interacted with the task table to load, select, and save each task. An example of the task section is illustrated in Fig. 15.

As mentioned in Section 4.1.1, the user study consisted of two types of tasks: free-form (type 1) and pre-determined (type 2) tasks. For the type 1 tasks, we selected specific

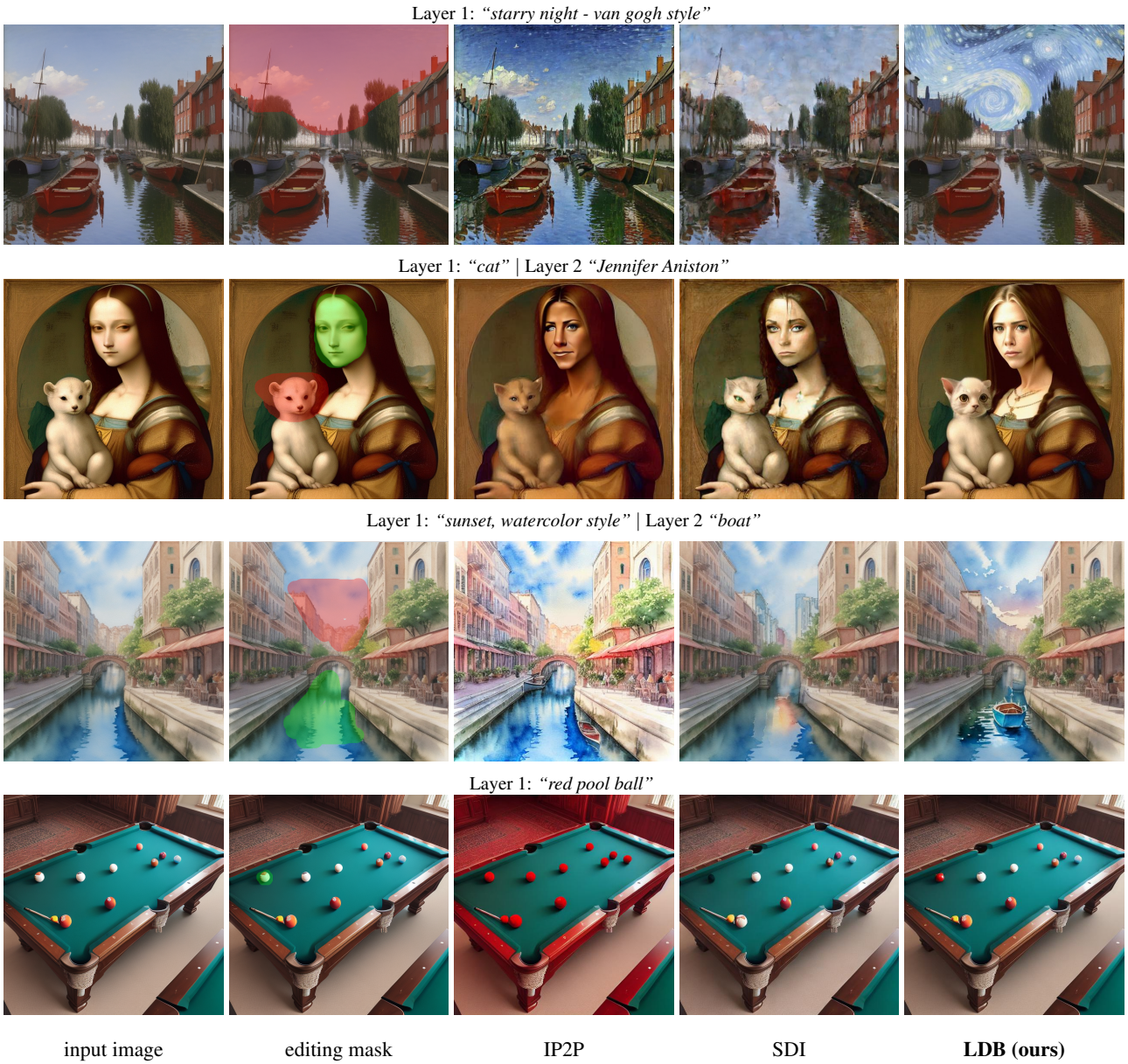


Figure 9. Qualitative results for the freeform part of the user study (Type 1 tasks)

types of edits that showcase various functionalities and capabilities of the system. Here are the description of edit types along with an example used during the user study:

1. Stack layers and create sequential edits (draw with LDB):
   - Input image: photo of a beautiful beach.
   - Layer 1: boat (Introduce a boat in the sea)
   - Layer 2: rocks (Scatter weathered rocks along the shoreline)
   - Layer 3: birds (Populate the sky above the boat with a flock of birds.
2. Modify attributes and features of objects:
   - Input image: portrait of a young man
   - Layer 1: blond (Transform a person's hair color to blond).
   - Layer 2: joker (Perform facial manipulation by swapping one person's face with another's, reshaping identities.)
3. Correct image imperfections and errors:
   - Input image: portrait of a man holding an umbrella
   - Layer 1: remove the rod that is mistakenly placed
   - Lyaer 2: fix the extra part on the side of the coat
4. Enhance discernibility of similar objects through modification:

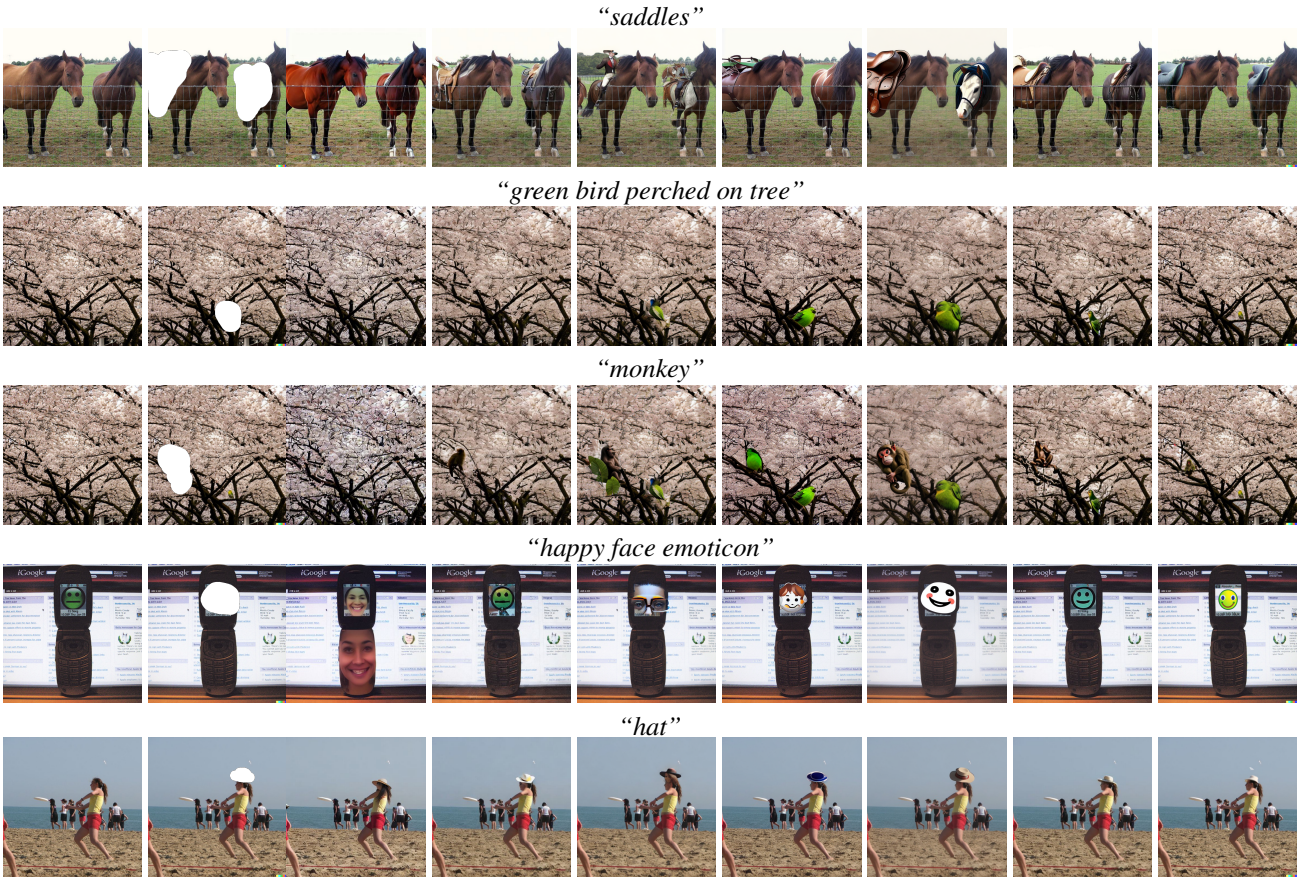


Figure 10. Additional qualitative examples on the MagicBrush dataset. Note that the images are not cherry picked and correspond to the user study (IP2P, SDI, LDB) and quantitative evaluation (BLD, HDP, BrushNet) with default settings.

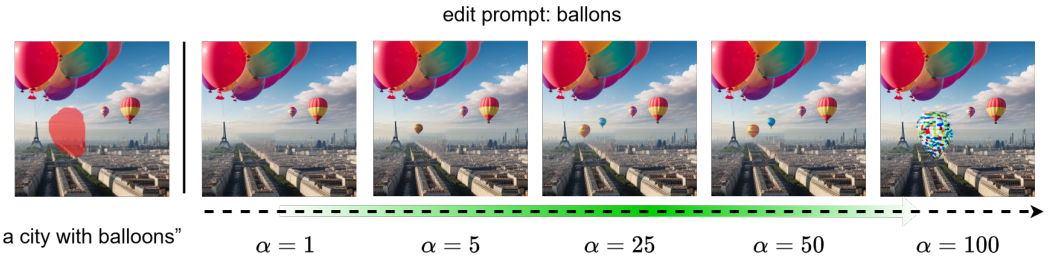


Figure 11. Ablation study on the effect of the strength parameter (α) in LDB. We incrementally increase the mask strength (α) while keeping the mask, seed, and intermediate denoising steps (n) fixed. A value of α that is too large introduces too much noise injection and may cause artifacts, while a value that is too small results in insufficient editing.

- Input image: aerial photo of a pool table with balls
- Layer 1: change the colour of a specific ball (third ball from the left) to red

5. Target specific regions for style transfer, refining aesthetics:
   - Input image: Mona Lisa by Leonardo Da Vinci
   - Layer 1: make the left part of the background similar to van gogh starry night style.

In our study design, we strategically chose the combination of seeds and prompts to encompass and evaluate these functionalities. Each user was given three seed-prompt items and tasked with creating and editing up to three layers of edits. For the majority of the tasks, N, i.e. the total number of steps for editing was set to $n = 5$. All the images

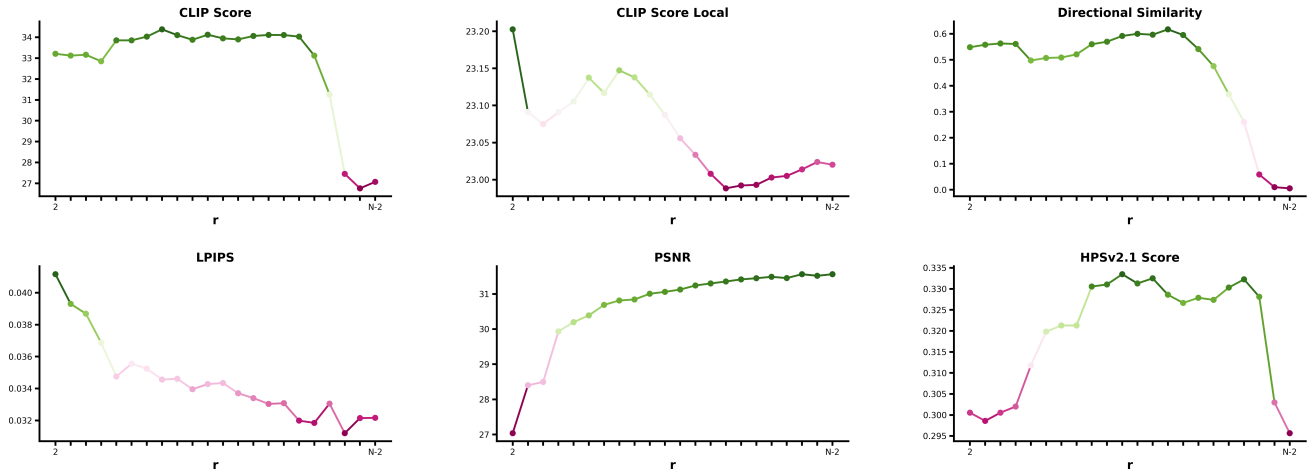


Figure 12. Quantitative evaluation of metrics across different regeneration step values (r). The x-axis represents the regeneration step r, increasing from left to right from 2 to $N-2$, while the y-axis shows the corresponding score values for each metric.

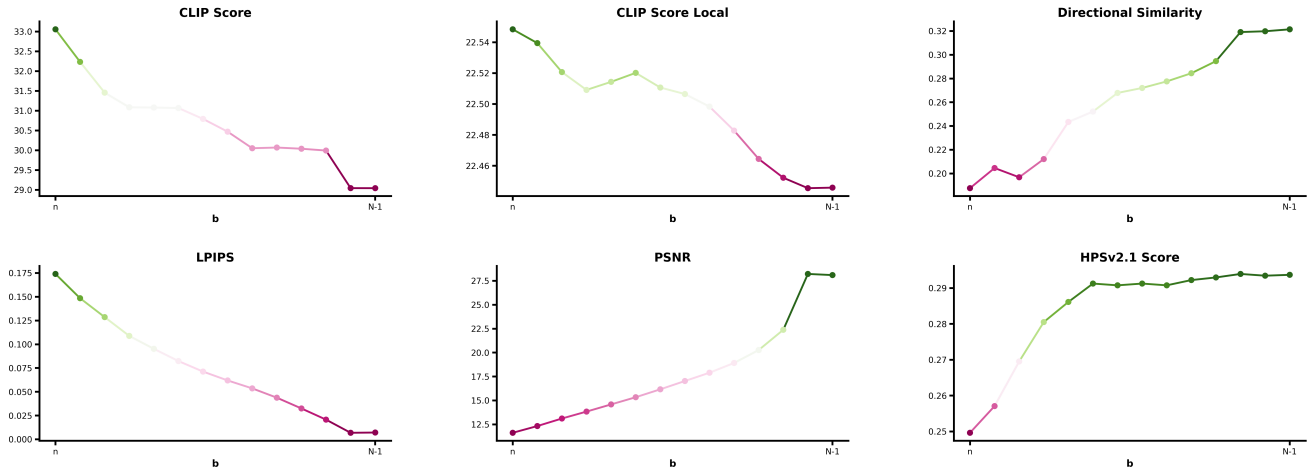


Figure 13. Quantitative metrics for different blending steps (b): The x-axis represents the blending step b, increasing from left to right from $b = n$ to N, while the y-axis shows the corresponding score values for each metric. Smaller b steps lead to poor background protection, while larger b values preserve background integrity and improve edit effectiveness.

were generated using Dreamshaper-7 [12] and the DDIM scheduler.

For the LDB method, users started by selecting a layer with an existing edit instruction from the task table, then created the corresponding layer in the UI. They had the option of choosing either the box option or the custom mask option. The task was followed by drawing the mask, tweaking the controls or edit prompt if needed, and completing the edit. Once the task was complete, the user saved the edit and moves on to the next task.

Users followed a similar procedure for the IP2P and SDI methods, with the exception of creating layers, as these methods do not incorporate layering capabilities. After completing each layer edit task, users saved the edits, and the user interface (UI) stacked subsequent edits onto the edited image. For IP2P method, users were required to write the instruction prompt and then adjust the image and text guidance scales and regeneration steps to finalize the edit. On the other hand, for the SDI method, users drew a mask and controlled the edit using the strength control. Completion times for each task were recorded for both methods.

Type 2 tasks, corresponding to the MagicBrush dataset [64], were more structured, with the mask, edit prompt, and input images provided by the dataset. MagicBrush utilized crowd workers to collect manual edits using DALL-E 2 [46]. This process involved 5,313 editing sessions and 10,388 editing iterations, resulting in a robust benchmark for instructional image editing. Additionally, the dataset provides manually annotated masks and instructions for each edit and contains up to three layers of edits. Users

Figure 14. Video editing examples using LDB and Stable Video Diffusion (SVD). The top row displays frames from an input video generated by SVD. For localized editing, we define a mask on the first frame and apply LDB edits to this initial frame. LDB's caching mechanism is then extended to the temporal dimension within SVD, enabling efficient propagation of edits across subsequent frames. This allows for the creation of multiple editing layers, and even non-sequential fast modifications to different parts of the video by revisiting and adjusting previous layers, while maintaining temporal coherence.

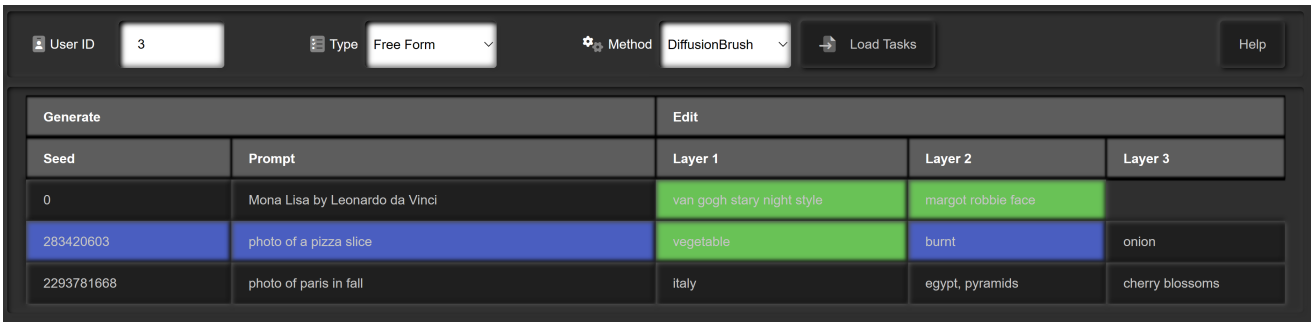


Figure 15. Overview of the tasks section, where users can interact to load, select, and save each task. Tasks that are selected are highlighted in blue, while those completed and saved are highlighted in green.

selected each image, started with the provided mask, could modify the mask if necessary, adjusted the control parameters and prompt, and saved and completed the task for each method.

10.2. Evaluation Survey

After completing the image editing tasks, the participants were asked to complete a three stage evaluation survey. The first part included a System Usability Scale (SUS) form to

rate the usability, ease of use, design, and performance of each method. SUS is a standard usability evaluation survey which is widely used in user-experience literature [8]. The participants were presented with 10 questions about each of the methods and were asked to rate each system on a scale of 1 to 5 for each question. A rating of 1 indicated strong disagreement, while a rating of 5 indicated strong agreement. The questions were designed to assess the participants' perceptions of the effectiveness, ease of use, and overall user experience of each tool. Below is the list of the questions:

Q1 I think that I would like to use this tool frequently.
Q2 I found the tool unnecessarily complex.
Q3 I thought the tool was easy to use.
Q4 I think that I would need the support of a technical person to be able to use this tool.
Q5 I found the various functions in this tool were well integrated.
Q6 I thought there was too much inconsistency in this tool.
Q7 I would imagine that most people would learn to use this tool very quickly.
Q8 I found the tool very cumbersome to use.
Q9 I felt very confident using the tool.
Q10 I needed to learn a lot of things before I could get going with this tool.

SUS consists of positive and negative phrasing questions. Q2, 4, 6, 8, and 10 are negatively framed, therefore on the chart, red colours means better SUS score and Q1, 3, 5, 7, and 9 are considered positively framed and hence, more green colours demonstrate better score.

The survey was followed by an interview with each participant to gather specific feedback and insights based on their artistic background and experience using the different tools. These processes provided valuable information on the strengths and weaknesses of each tool, as well as how it can be improved to better serve users.

The following multiple-choice questions were also asked for evaluating the performance of each method:

- How much time did it take you to complete the image editing task using the tool you used in this study? [Much less time/About the same/Much more time]
- How did you find each of the tools in terms of effectiveness in achieving the desired edits? [Very effective/Somewhat effective/Neutral/Somewhat ineffective/Very ineffective]
- How does each of the tools you used perform in terms of time to complete the editing task? [Much faster/Somewhat faster/Acceptable/Somewhat slower/Much slower]
- How likely are you to use each of these tools as an AI image editing tool in the future? [Very likely/Somewhat likely/Neutral/Somewhat unlikely/Very unlikely]

The entire study, including filling out the evaluation surveys, took not more than 90 minutes.

Fig. 20 illustrates the outcomes of the post-study CSI survey. Overall, participants expressed positivity towards LDB, indicating that it enhanced their enjoyment, exploration, expressiveness, and immersion, while also deeming the results worth their effort. The CSI score results also show that one participant responded neutrally or negatively to certain aspects, likely due to their being accustomed to the Photoshop tool. Furthermore, there was notable variability in immersion scores, with several participants giving lower ratings. This variability suggests that while some users felt deeply engaged with the tool, others may have encountered challenges or distractions affecting their immersive experience. Analyzing specific factors such as interface design, task complexity, and user preferences could offer insights into enhancing immersion in future iterations of LDB. Despite this variability, the majority of participants found the tool effective and engaging, highlighting its potential usefulness in creative workflows.

One of the most common comments regarding the usability of different methods was that participants found it challenging to find the optimal settings for IP2P and SDI. For example, one user mentioned, "In InstructPix2Pix, increasing the image guidance scale often distorts the edited image too much, and if the text guidance scale is too high, the edited image looks completely different. After many trials and errors, when I find a good combination, the next image behaves differently. Also, SD-inpainting half the times fails to produce a satisfactory result."

Another user, who is an expert in graphic design, suggested, "Layers are very helpful. I would like to see the control numbers on top of them as I change them, not beside them. Also, having an undo button is crucial and would be very helpful. Additionally, I would suggest adding a blend option to each layer, similar to Photoshop". These suggestions will be taken into consideration for future improvements.

10.3. System Usability Scale (SUS)

Fig. 19 presents the results of the SUS survey among participants after using LDB, SDI, and IP2P. Based on the bar charts, participants indicated that they are more likely to use LDB compared to IP2P and SD-Inpainting, and that they find it the easiest tool to use. In addition, participants in Q4 expressed that they would not require technical assistance to use the system in the future, indicating its overall good design. These findings were further supported by the interview feedback. For example, when asked about their understanding of the different parameters in the tool, one participant stated: "I believe that I understand the functionality of each parameter. I need to increase the mask strength value if I want to make bigger changes. The tool is quite intuitive and easy to use, and I think I can easily use it without needing any technical support." This feedback highlights

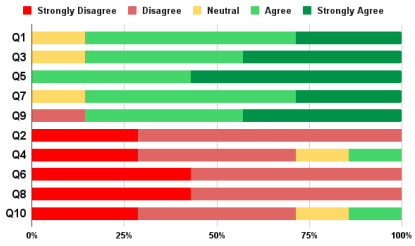


Figure 16. LDB usability

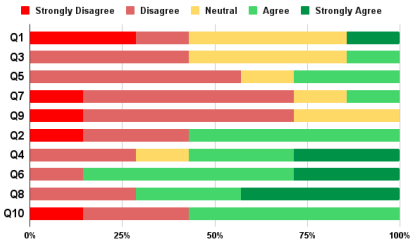


Figure 17. SDI usability

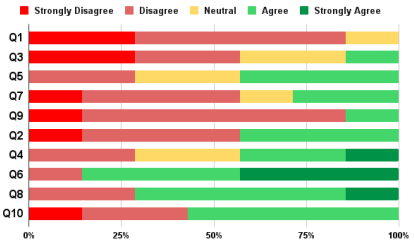
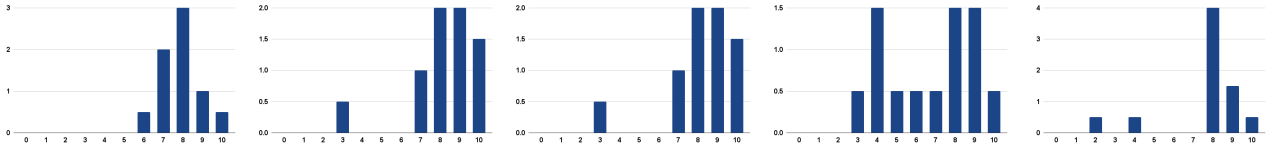


Figure 18. IP2P usability

Figure 19. Results of Q1 - Q10 for the usability of each system among different participants. For odd questions, green colors show more desirable feedback. Even questions are designed with negative wording and more red colors show more favorable feedback.



Enjoyment Expressiveness Exploration Immersion Result Worth Effort

Figure 20. Histogram of the Creativity Support Index from the user study survey.

that the tool has a user-friendly design and can be easily understood and used by a wide range of users. Based on the survey results, the SUS score for LDB is calculated as 80.35%, while IP2P and SDI achieve a score of 38.21% and 37.5% respectively.

For CSI [10] questionnaire we used all questions, excluding questions about collaboration as it is not relevant for our tool. The CSI measures dimensions of Exploration, Expressiveness, Immersion, Enjoyment, and Results Worth Effort in a tool. CSI helps in understanding how well LDB support creative work overall, as well as pointing out which aspects of creativity support may need attention.

Figure 20 illustrates the outcomes of the post-study CSI survey. Overall, participants expressed positivity towards LDB, indicating that it enhanced their enjoyment, exploration, expressiveness, and immersion, while also deeming the results worth their effort.

11. Initial User Study Insights

We initially developed an earlier version of LDB, called *Diffusion Brush*, with the objective of re-randomizing targeted regions for fine-tuning (*e.g.* fixing small details that were generated incorrectly) and without layering functionalities. Subsequently, we conducted a user study to assess its usability and features and based on the feedback received from this first study, we made significant improvements and revamped the tool. In the first user study, we compared the early version of LDB with SDI and manual editing in Adobe Photoshop [26], involving five expert users.

While the majority of participants acknowledged that Diffusion Brush was faster than manual editing, some participants suggested that even faster editing would be significantly beneficial, aiding in random idea generation for artists. To address this feedback, we incorporated a caching mechanism, as explained in Section 3.1, designed an efficient front-end to communicate with the machine learning backbone, and highly optimized the overall pipeline, achieving as little as 140 ms of inference time for a single edit on a high-end consumer GPU.

Furthermore, a few users struggled with finding the optimal brush strength control, a similar challenge observed in SD-inpainting as well. To address this, we devised a more generalized approach. While the earlier version of our system also supported multiple masks, these masks were not fully independent, and deleting or hiding them was not possible without performing operations in a specific order. This observation prompted the creation of a more streamlined and flexible mask management system.

Additionally, insights gathered from the first round of interviews indicated the need for further improvement in various aspects of the tool's functionality and user experience. These inputs guided us in refining the tool and enhancing its usability for a wider range of users. Lastly, in the first user study, three participants specifically mentioned this feedback. One participant stated, "I really like the tool as it is right now; it certainly provides value for me in my editing tasks and makes my life easier. But one feature that I would love to see is to be able to tell the system how to make these

changes. I still want to use the masking editing, but if I can tell it what to do it would be great.". Based on the findings of the new user study, it is evident that this feature has been well-implemented into the system. All users participating in the current study affirmed the effectiveness of this feature.