FlowEdit: Inversion-Free Text-Based Editing Using Pre-Trained Flow Models
(Reproduction Report)

Mohamed Jalal Baim, Ayoub Amine

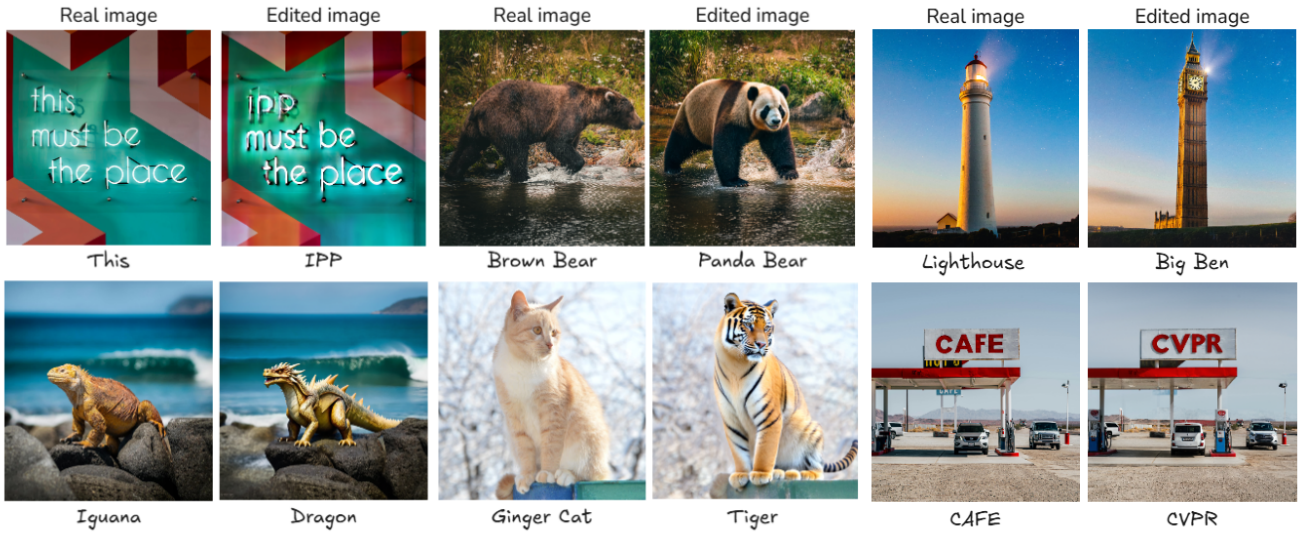

Figure 1. **FlowEdit.** Given a real input image and a target text prompt, FlowEdit performs precise semantic edits while preserving the original structure and identity. Across diverse editing scenarios, the method maintains background consistency, object pose, and fine details, demonstrating strong structure preservation alongside faithful adherence to the target prompt.

Abstract

Editing real images with natural language prompts remains a central challenge. In this report, we present ***FlowEdit***[1]*, an inversion-free, optimization-free, and model-agnostic method for text-based image editing built on pre-trained text-to-image flow models.*

Given a real input image and a target text prompt, the objective is to modify the image so that it satisfies the new description while preserving its original structure. FlowEdit proposes a novel approach that constructs an ODE directly connecting the source and target image distributions. Rather than mapping images to noise and back, the method defines a continuous transformation that transports the source distribution to the target distribution conditioned on the desired prompt.

We evaluate the method using quantitative metrics and compare it to existing approaches, demonstrating improved structure preservation and adherence to the target prompt.

[1]Pan, Z., et al. (2024). "FlowEdit: Generalizable Flow-Based Image Editing." CVPR. https://arxiv.org/abs/2403.11978

1. Introduction

Text-based image editing aims to modify a real image using a text prompt. Most existing approaches follow an editing-by-inversion paradigm: the input image is first mapped to a latent noise representation using a pre-trained generative model, and then regenerated under a target text prompt. Although effective in some cases, this procedure requires traversing a long trajectory through the noise space. In practice, this often leads to structural distortions, visual artifacts, and unstable behavior, particularly for strong edits. Even with perfect inversion, this long path through noise causes loss of structure.

FlowEdit is motivated by a simple question: *can we perform text-based image editing without passing through noise at all?* To address this, FlowEdit introduces an inversion-free, optimization-free, and model-agnostic framework for editing with pre-trained text-to-image flow

models. Instead of reconstructing the image, FlowEdit constructs a direct editing trajectory between the source and target distributions.

In this report, we detail the FlowEdit methodology and present experiments conducted using our own implementation of the algorithm, including both quantitative and qualitative evaluations. The full implementation is publicly available on GitHub[2].

2. Methodology

To perform text based image editing, let's assume we have a real image X_{src}, a source text prompt describing this image, and a target text prompt describing the desired output. This output should be the edited image X_{tar} that follows the target prompt but still looks like the original image.

FlowEdit allows editing without inversion, optimization at test time, or architectural changes. Unlike inversion-based methods, FlowEdit moves X_{src} from the source distribution to the target distribution via a continuous process (velocity field).

To illustrate this idea, Flow-based generative models define data evolution using a velocity field. FlowEdit starts from X_{src} and applies a sequence of small updatesthat slowly change the image so that it matches the target text, while keeping the original structure. At each time step t, FlowEdit computes:

- How the model would move the image if it followed the source prompt: $V_{\text{src}}(Z_t^{\text{src}}, t)$.
- How the model would move the image if it followed the target prompt: $V_{\text{tar}}(Z_t^{\text{tar}}, t)$.

The difference between these movements defines the editing direction used to update the image (see Figure 2).

2.1. Algorithm

Algorithm 1 Simplified algorithm for FlowEdit

Input: real image X^{src}, $\{t_i\}_{i=0}^{T}$, $n_{\max}$, n_{avg}
Output: edited image X^{tar}
Init: $Z^{\text{FE}}_{t_{n_{\max}}} = X^{\text{src}}$
for $i = n_{\max}$ **to** 1 **do**
  $N_{t_i} \sim \mathcal{N}(0, 1)$
  $Z^{\text{src}}_{t_i} \leftarrow (1 - t_i)X^{\text{src}} + t_i N_{t_i}$
  $Z^{\text{tar}}_{t_i} \leftarrow Z^{\text{FE}}_{t_i} + Z^{\text{src}}_{t_i} - X^{\text{src}}$
  $V^{\Delta}_{t_i} \leftarrow V^{\text{tar}}(Z^{\text{tar}}_{t_i}, t_i) - V^{\text{src}}(Z^{\text{src}}_{t_i}, t_i)$
  $Z^{\text{FE}}_{t_{i-1}} \leftarrow Z^{\text{FE}}_{t_i} + (t_{i-1} - t_i)\, V^{\Delta}_{t_i}$
end for
Return: $Z^{\text{FE}}_0 = X^{\text{tar}}$

More details on the algorithm, Theoretical background and justification are available in Appendix 1.1.

[2] github.com/Jalalbaim/FlowEdit-implementation

3. Experimental Evaluation

Dataset. The experiments from the paper were reproduced using images from the DIV2K dataset. For the quantitative evaluation, and due to computational issues, only 10 images were used (compared to 70 in the original work), each with a resolution of $512 * 512$ pixels instead of $1024 * 1024$.

Implementation Details. In our experiments, we used only the official weights of SD3, available on Hugging Face, instead of SD3 and Flux as employed in the original work. The hyperparameters were set to $T = 50$ steps and $n_{max} = 33$. Since SD3 uses Classifier-Free Guidance (CFG) for text conditioning, the source and target scales were set to 3.5 and 13.5, respectively.

Methods compared. We compare FlowEdit with T2I (text-to-image) editing approaches for flow models, using only SD3 (instead of SD3 & FLUX as stated in the original paper). Specifically, we evaluate SDEdit by adding noise to the source image up to step n_{max} before sampling with the target prompt.

Here is a summary table of all hyperparameters used in these experiments:

Table 1. Hyperparameters of compared methods.

	T steps	$n_{\max}$	CFG @src	CFG @tar
SDEdit	50	10, 15 ... 40	–	13.5
FlowEdit	50	33	3.5	13.5, 16.5, 19.5

3.1. Results

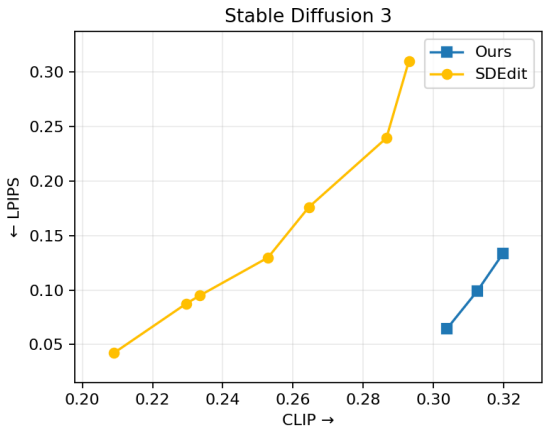

Figure 3. **Quantitative results.** Trade-off between CLIP (higher is better) and LPIPS (lower is better) on SD3.

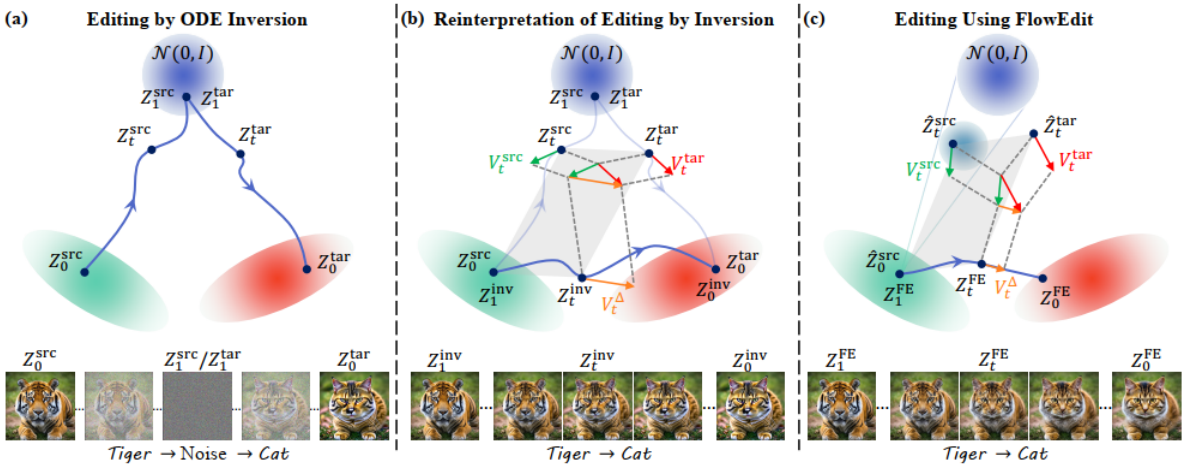

Figure 2. (Figure 2 from paper) (a) In inversion-based editing, the source image is first transformed into a noise representation, and then reconstructed using the target prompt. (b) This process can also be seen as a path that directly moves the image from the source to the target by following the changes predicted by the model. (c) FlowEdit directly updates the image step by step. At each step, it adds a small amount of random noise to estimate how the model would change the image for the source prompt and for the target prompt. The difference between these two gives the direction of edits, which is then used to update the image.

Quantitative. Figure 3 shows the trade-off between text adherence (CLIP, higher is better) and structure preservation (LPIPS, lower is better) for different methods on SD3. FlowEdit consistently achieves lower LPIPS values for higher CLIP scores than SDEdit.This indicates that FlowEdit preserves the structure of the source image more effectively while still respecting the target text. In contrast, SDEdit can only maintain low LPIPS values when text adherence is weak, and stronger edits lead to a significant degradation of structural consistency.

Qualitative results. Figure 1 presents editing results from proper reimplementation of FlowEdit across diverse images and target prompts, including text editing and object modifications. In all examples, FlowEdit produces edits that follow the target instructions while preserving the global structure of the source image. Also, object-level transformations such as *Iguana* → *Dragon* or *Ginger Cat* → *Tiger* retain consistent pose, lighting, and background context, indicating strong structural preservation.

A qualitative comparison between FlowEdit and SDEdit is available in Appendix 2.1, also an ablation study on the importance of each hyperparameter $n_{max}, nmin$ is provided in Appendix 2.2, 2.3.

4. Discussion

Our reproduction confirms that FlowEdit offers a superior balance between text adherence and structure preservation compared to inversion-based methods like SDEdit. While SDEdit forces a trade-off between content and structure, FlowEdit achieves both via direct transport.

However, FlowEdit also comes with some limitations as shown in Appendix 2.4. Since it is mainly designed to keep the original image structure, it struggles handling large or complex edits such as changing the background, or adjusting object poses. Due to its intrinsic property, the method tends to follow a safe and careful path, avoiding big changes.

In order to adress these limitations, our idea was to introduce a spatially decoupled flow. Inspired from DiffEdit's paper, the main idea is to "divide" the image into two regions, one region that includes pixels that should be edited and the other one pixels that should preserve the image identity.

This will be discussed in detail in the second part, where we will provide insights how to adapt DiffEdit's approach to FlowEdit and discuss some results.

5. Adressing FlowEdit's Limitations (Part 2)

Couairon et al. proposed DiffEdit[3], a method designed for semantic image editing using text-conditioned diffusion models without requiring manual user masks. It operates on the principle that a diffusion model will yield different noise estimates when conditioned on different text prompts (e.g., "horse" vs. "zebra").

By contrasting these predictions, DiffEdit can automatically identify the specific regions of the image that need to be modified to match a target query. Then push the model

[3]Couairon, G., et al. (2023). "DiffEdit: Diffusion-based semantic image editing with mask guidance." ICLR. https://arxiv.org/abs/2210.11427

to focus its edits only on these relevant regions so they guarantee strong edits as well as structure preservation.

The main challenge we encounter with FlowEdit is that its intrinsic ability to preserve the original image identity hinders its capacity to apply complex edits. To address this, we propose giving the model more freedom in the targeted regions while constraining it in the non-targeted areas.

Following this reasoning, we suggest that the Flow model will also generate distinct velocity fields when conditioned on different text prompts. By contrasting these velocity estimations, we can derive a binary mask that separates the image into targeted and non-targeted regions.

The targeted regions follow the target velocity field (V_{tar}), granting the model the semantic freedom to modify complex structures according to the new prompt. Meanwhile, the non-desired regions are anchored to the source velocity field (V_{src}), which forces the model to maintain the original image's identity and background. The pipeline 4 shows the whole approach in details.

At the end we tend to solve a direct ODE path where the update direction at each step is a weighted average of two velocities. By applying the mask M, we define the blended velocity as $V_{blend} = M \cdot V_{tar} + (1 - M) \cdot V_{src}$. This ensures that the flow matching process is only "freed" to change pixels within the masked area.

Henceforth, we call this new approach FocusFlow.

5.1. FocusFlow Algorithm

Algorithm 2 Simplified FocusFlow Algorithm

Input: real image X^{src}, source prompt p_{src}, target prompt p_{tar}, $\{t_i\}_{i=0}^{T}$, $n_{\max}$, M mask.
Initialize: $z_t^{edit} \leftarrow x_{src}$
for $i = n_{\max}$ **to** 1 **do**
  $\epsilon \sim \mathcal{N}(0, I)$
  $z_t^{src} \leftarrow (1 - t_i)x_{src} + t_i\epsilon$
  $z_t^{tar} \leftarrow z_t^{edit} + z_t^{src} - x_{src}$
  Compute velocities (V_t^{src}, V_t^{tar})
  $V_{blend} \leftarrow M \cdot V_t^{tar} + (1 - M) \cdot V_t^{src}$
  $z_t^{edit} \leftarrow z_t^{edit} + (t_{i-1} - t_i)V_{blend}$
end for
return $(x_{out}, M_{soft}, M_{bin})$

5.2. Experiments

We conducted our experiments using the same framework as FlowEdit, with a specific emphasis on analyzing failure cases, particularly those involving background distortions and pose editing inaccuracies.

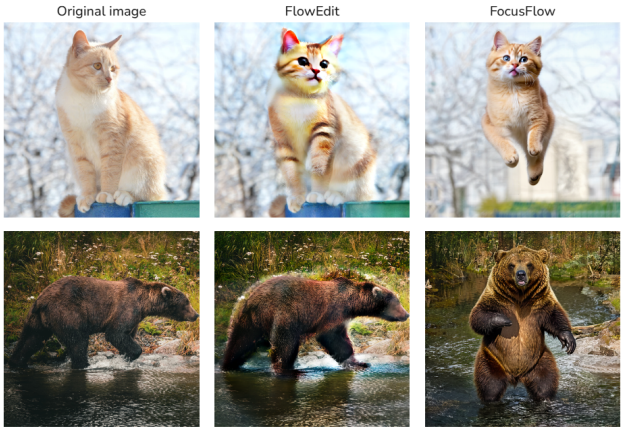

Figure 5. Qualitative results. The first row shows the editing results of cat $sitting \rightarrow jumping$, the second row shows the results of bear $walking \rightarrow standing$.

Based on our findings in 5, we see that FocusFlow significantly outperforms FlowEdit in localized semantic image editing, as demonstrated across cat and bear transformations.

6. Conclusion

As highlighted in the FlowEdit paper, the method shows clear limitations when handling large or complex edits. Drawing inspiration from DiffEdit, originally proposed for diffusion-based models, we adapted its two-stage methodology to automatically generate a binary mask that separates the image into targeted and non-targeted regions. The algorithm then solves the associated ODE, where the update direction is defined by a blended velocity field: a strong velocity V_{tar}, applied to targeted regions and a more conservative one V_{src}, applied to non-targeted areas.

Quantitatively, by leveraging this velocity blending strategy, FocusFlow achieves robust editing capabilities, particularly in challenging tasks such as pose modification, an area where FlowEdit struggles due to its delta-based design. Specifically, FlowEdit relies on a velocity difference $V^{\Delta} = V^{\text{tar}} - V^{\text{src}}$, which tends to be overly conservative and only effective for precise and local edits.

To sum up, this work exposes FlowEdit's limitations and addresses them through a new approach inspired by DiffEdit, which we introduce as FocusFlow.

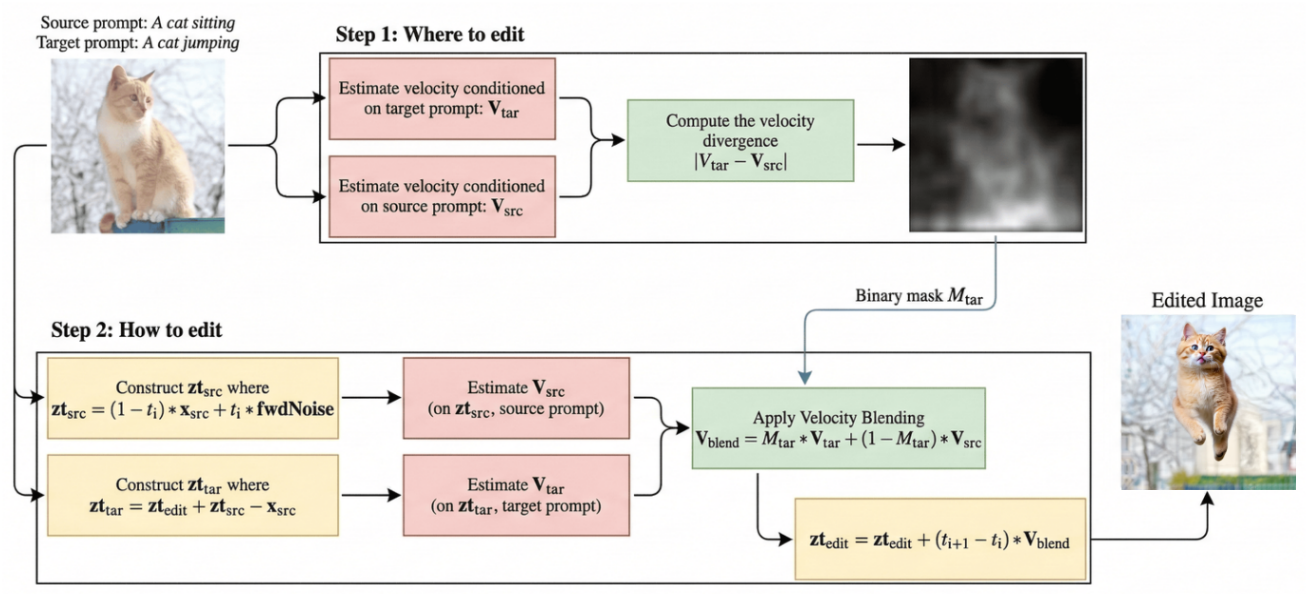

Figure 4. FocusFlow Pipeline Overview. Two-stage editing process: (1) Mask generation M based on velocity difference; (2) Image editing $V = M * V_{tar} + (1 - M) * V_{src}$ solves the ODE for region specific edits.

FlowEdit: Inversion-Free Text-Based Editing Using Pre-Trained Flow Models

Supplementary Material

BAIM Mohamed Jalal, AMINE Ayoub

1. FlowEdit Method

To explain the motivation behind the method, let us focus on a simple two-dimensional example showed in Figure 6, in which the source and target distributions are Gaussian mixtures with two modes each.

The inversion approach maps samples from the source distribution into their initial Gaussian noise distribution by inverting the flow process with the source prompt.Then it employs a sampling process with the target prompt, transporting the samples into the target distribution. As a result, we notice this approach can lead to intermixed modes. In contrast, the path taken by FlowEdit give rise to not mixed modes.

Actually, during the inversion, a point is mapped to noise, which is closer to the more distant mode. This point will be driven to the mode closest to it, resulting in an unnecessarily long editing path.

On the other hand, if we take the same point, FlowEdit constructs a path that starts at $T = 1$ and ends at $T = 0$. Shortly after we start at $T = 1$, we pause and go through the algorithm. To determine the motion direction, FlowEdit starts by adding noise to the source image, this is done by sampling random noise N_t, and combining it with the source image. Then the flow model is applied on each noisy version of the source image using the source prompt, and calculate the average direction. Next, we use the vector between the source image and our current edited image to construct noisy versions of the target image. We apply the model on each noisy target image with the target prompt and calculate the average direction. The difference between the two vectors, V_Δ, drives our ODE between the closest source and target modes.

These steps are repeated until we reach the target.

1.1. Theoretical justification and guarantees

FlowEdit is based on the theory of rectified flow models, where data generation is maintained by an ordinary differential equation (ODE).

$$\frac{dZ_t}{dt} = V(Z_t, t), \qquad t \in [0, 1],$$

trained such that the marginals satisfy

$$Z_t \sim (1 - t)X_0 + tX_1, \qquad X_1 \sim \mathcal{N}(0, I).$$

This property guarantees that all intermediate states lie on the data noise manifold seen during training.
For text-based editing, let the velocity the vector field that tells how an image representation moves over time along the generative path.

$$V^{\text{src}}(x, t) = V(x, t, c_{\text{src}}), \qquad V^{\text{tar}}(x, t) = V(x, t, c_{\text{tar}}),$$

and consider inversion-based editing defined by

$$\frac{dZ_t^{\text{src}}}{dt} = V^{\text{src}}(Z_t^{\text{src}}, t), \; Z_0^{\text{src}} = X_{\text{src}},$$

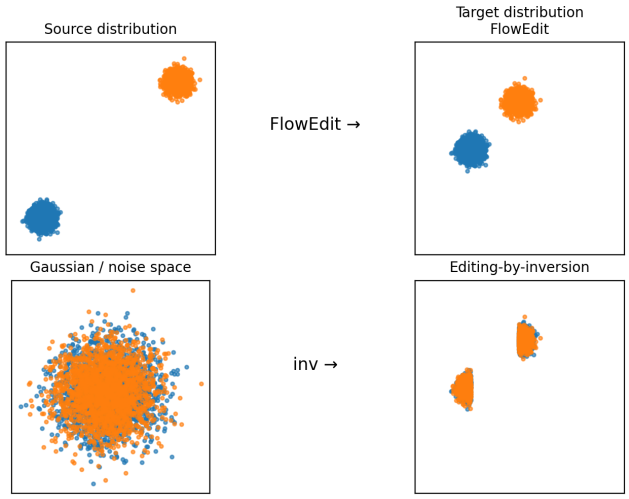

Figure 6. Source to target pairings for editing-by-inversion and FlowEdit. Source and target distributions are Gaussian mixtures with two modes each.

$$\frac{dZ_t^{\text{tar}}}{dt} = V^{\text{tar}}(Z_t^{\text{tar}}, t), \; Z_1^{\text{tar}} = Z_1^{\text{src}}.$$

The paper shows that this procedure implicitly defines a direct path

$$Z_t^{\text{inv}} = Z_0^{\text{src}} + Z_t^{\text{tar}} - Z_t^{\text{src}},$$

satisfying $Z_1^{\text{inv}} = X_{\text{src}}$ and $Z_0^{\text{inv}} = Z_0^{\text{tar}}$. So by differenting, we get the direct ODE

$$\frac{dZ_t^{\text{inv}}}{dt} = V^{\text{tar}}(Z_t^{\text{tar}}, t) - V^{\text{src}}(Z_t^{\text{src}}, t).$$

Due to the shared noise content of Z_t^{src} and Z_t^{tar}, the velocity difference removes noise components.
FlowEdit generalizes this path by replacing Z_t^{src} with a random process,

$$\hat{Z}_t^{\text{src}} = (1 - t)Z_0^{\text{src}} + tN_t, \qquad N_t \sim \mathcal{N}(0, I),$$

and defining $\hat{Z}_t^{\text{tar}} = Z_t^{\text{FE}} + \hat{Z}_t^{\text{src}} - Z_0^{\text{src}}$. The resulting ODE is

$$\frac{dZ_t^{\text{FE}}}{dt} = \mathbb{E}\left[V^{\text{tar}}(\hat{Z}_t^{\text{tar}}, t) - V^{\text{src}}(\hat{Z}_t^{\text{src}}, t) \,\middle|\, Z_0^{\text{src}}\right], \qquad Z_1^{\text{FE}} = Z_0^{\text{src}}.$$

This construction guarantees in-distribution model queries and defines a well-posed transport ODE. While FlowEdit does not guarantee optimal transport or exact matching of the target distribution, it induces a shorter transport path than inversion, explaining its improved structure preservation.

2. Experimental Evaluation

2.1. FlowEdit vs. SDEdit

Figure 7 provides a direct qualitative comparison between FlowEdit and SDEdit. While SDEdit often struggles to simultaneously enforce the target prompt and maintain the original structure, FlowEdit achieves both. For instance, in the second row, FlowEdit successfully transforms the lighthouse into "Big Ben" clock, whereas SDEdit fails to achieve the same level of fidelity.

Figure 7. **Qualitative comparisons.** With SD3 (with $n_{max} = 30$), FlowEdit adheres to the target prompt better than SDEdit.

Figure 8 illustrates the behavior of SDEdit under increasing edit strength compared to FlowEdit on the same prompt. As the SDEdit strength (n_{max}) increases, the model progressively enforces the target semantics, but this comes at the cost of severe structural drift: pose, proportions, and background consistency are gradually lost. In contrast, FlowEdit achieves a faithful tiger transformation while preserving the original pose, and scene background. To sum up, FlowEdit bridges this SDEdit limitation successfully, maintaining structural coherence even for semantically significant edits.

2.2. Impact of n_{max}

$n_{\max}$ controls the starting point of the FlowEdit ODE and therefore directly determines the edit strength: larger values induce stronger semantic changes. As shown in Figure 9, for $n_{\max} = 10$, the edit is minimal and the image remains very close to the original. At $n_{\max} = 30$, target semantics start to emerge while the global structure and layout are preserved. With $n_{\max} = 50$, the full path is traversed, resulting in a strong edit where target semantics dominate and structure preservation degrades.

2.3. Text-based style editing

A more challenging task for T2I editing methods, is text based style editing. As shown in Figure 10, FlowEdit demonstrates huge ability to perform accurate stylistic modifications.

In order to achieve such results, we tuned FlowEdit's hyperparameters, mainly $n_{\min}$. Actually, $n_{\min}$ controls the amount of deviation from the source image at low noise levels by determining when FlowEdit stops enforcing structure preservation.

For timesteps $i \leq n_{\min}$, the algorithm switches to standard sampling conditioned on the target prompt, allowing stronger modifications of high-frequency details.
As a result, increasing $n_{\min}$ enables stronger style and texture changes, at the cost of reduced fidelity to the original image, and then we end up using $n_{\min} = 15$.

2.4. Failed examples

FlowEdit is great at preserving image structure, but this also limits its ability to make big changes to large areas of the image. This is shown in Fig 11 for pose and background editing. In these cases, FlowEdit does not fully change the image according to the target prompt and fails to add the new desired elements. To get bigger changes from the source image, We tried to increase $n_{\min}$, as we've shown previously in Fig 10. Actually this works with style changes, but it is still not enough for background or pose edits.

2.5. GPU computation

To reproduce the results of the paper and run the experiments presented, We relied on multiple cloud GPU providers, mainly Kaggle and Google Colab. On Kaggle, experiments were conducted using an NVIDIA P100 GPU, while on Colab an NVIDIA L4 GPU was used. Due to computational constraints, We could only use 10 images per experiment with a single prompt, also experiments were limited to Stable Diffusion 3. The FLUX model is significantly more demanding and We could not execute it reliably on the available GPU resources.

Table 2 summarizes the computational resources and runtimes used for these experiments.

Platform	GPU	Memory	Runtime
Kaggle	NVIDIA P100	16 GB	30 hours
Google Colab	NVIDIA L4	22 GB	10 hours

Table 2. Summary of GPU resources and computation times used for the experiments.

Figure 8. **Effect of edit strength on structural preservation.** Comparison between SDEdit and FlowEdit for the prompt: *An orange and white cat sitting* $\rightarrow$ *A tiger sitting*.

Figure 9. Impact of $n_{\max}$

Figure 10. Text-based style editing results

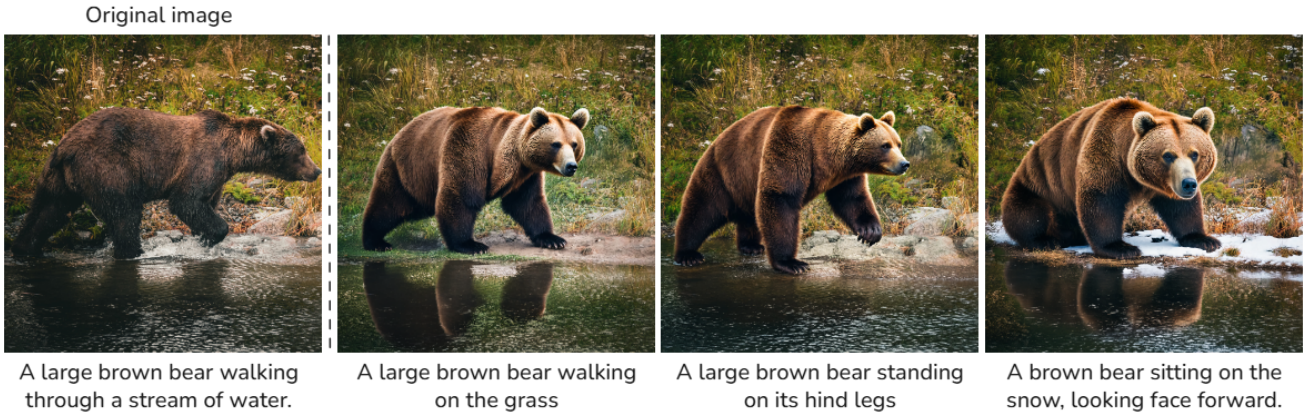

Figure 11. Some unsuccessful examples