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

Vladimir Kulikov Matan Kleiner Inbar Huberman-Spiegelglas Tomer Michaeli
Technion – Israel Institute of Technology

Contents

A Additional results **2**

B Comparisons **3**
B.1. Additional qualitative comparisons . . . 3
B.2. Additional details on the experiment settings . . . 4
B.2.1. Stable Diffusion 3 hyperparameters . . . 4
B.2.2. FLUX hyperparameters . . . 4
B.3. Metrics comparisons . . . 6

C Effect of $n_{\max}$ **7**

D Text-based style editing **8**

E Hyperparmeters used for Fig. 1 **9**

F. Full Algorithm **10**

G Limitations **11**

H Effect of random noise on the results **12**

I. Effect of source prompt on the results **13**

J. Illustration of the noise-free path between the source and target distributions **14**

K Effect of n_{avg} **20**

L Further discussion on the relation to optimization based methods **21**
L.1. Delta denoising loss . . . 21
L.2. FlowEdit with different step sizes . . . 22
L.3. Mathematical discrepancies between the FlowEdit, DDS, PDS update steps . . . 23

MAdditional details about the Cats-Dogs experiment **24**

A. Additional results

Figure S1 shows additional editing results obtained with FlowEdit.

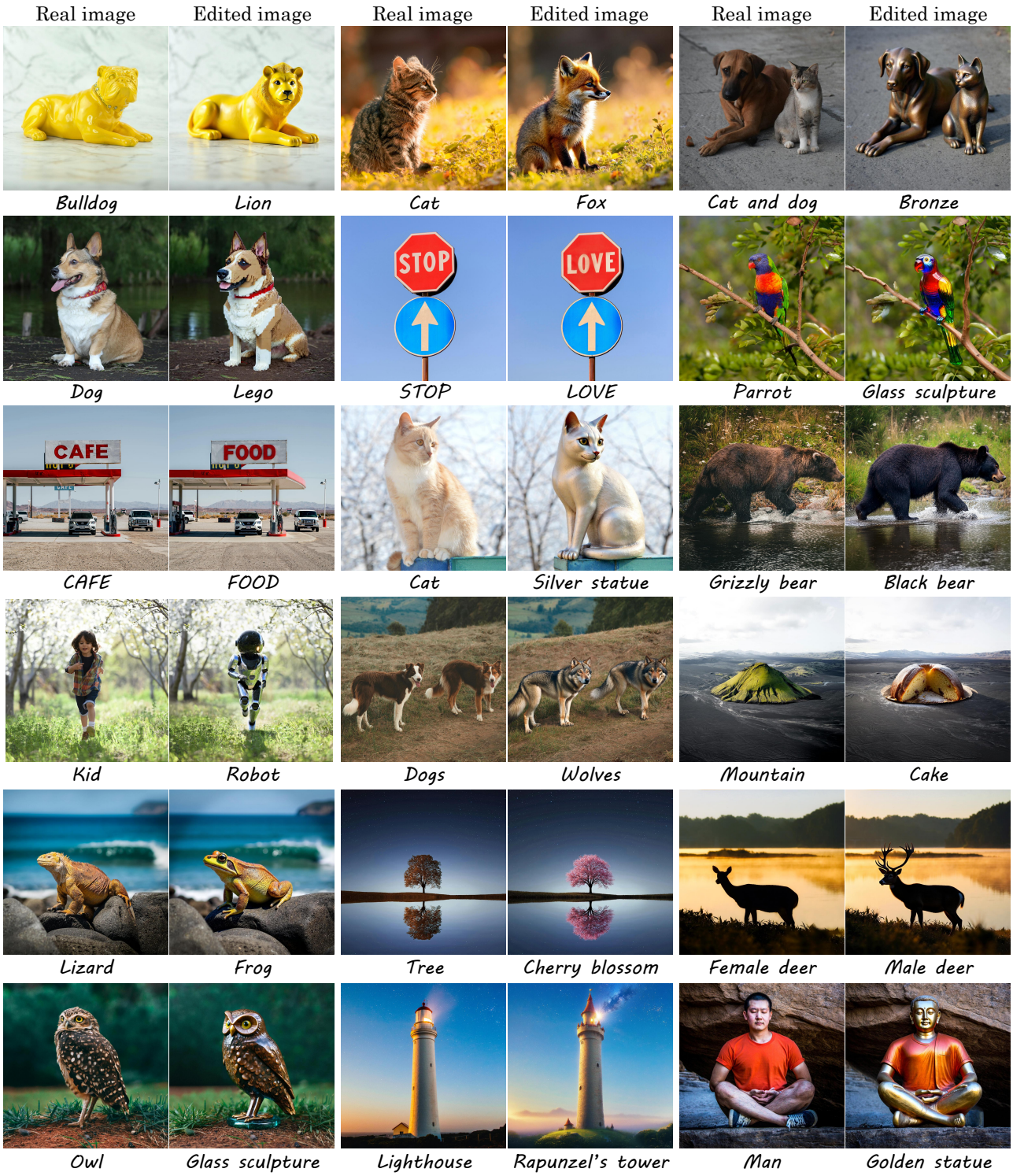

Figure S1. **Additional FlowEdit results.** FLUX was used for the first, third, and fifth rows, and SD3 for the second, fourth and sixth rows.

B. Comparisons

B.1. Additional qualitative comparisons

Figure S2 shows additional comparisons between FlowEdit and the competing methods with both SD3 (left) and FLUX (right). The value next to SDEdit indicates the editing strength (see App. B.2).

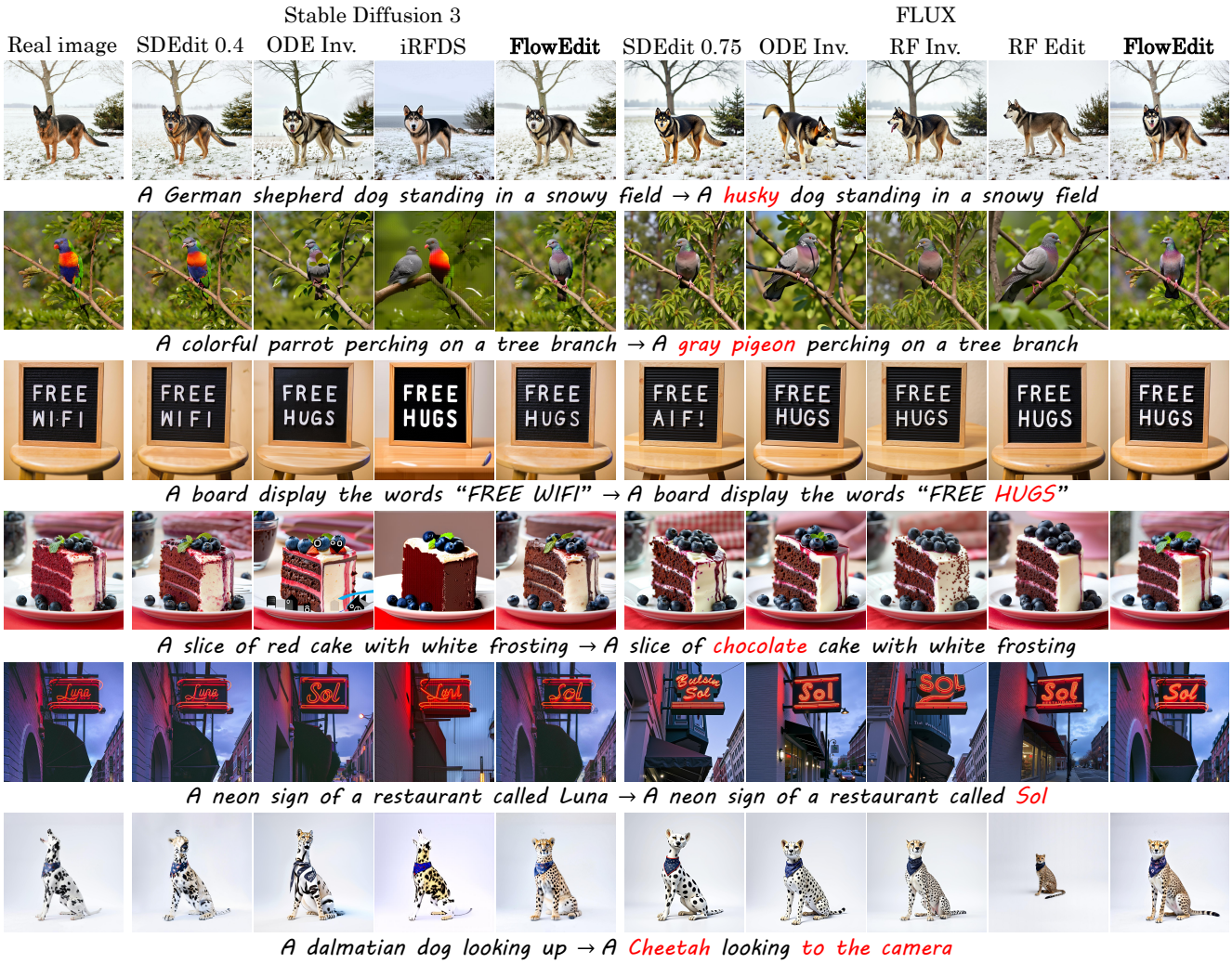

Figure S2. **Additional qualitative comparisons.**

B.2. Additional details on the experiment settings

We compare FlowEdit against the baseline methods of ODE inversion (Sec. 3.2) and SDEdit using both FLUX and SD3, as well as against RF-Inversion, RF Edit (FLUX) and iRFDS (SD3). We were not able to compare to Stable Flow [1], as their method is not compatible with Nvidia RTX A6000 (which we used for running FlowEdit, as well as all the other methods) even with cpu offloading and when changing the image size from 1024×1024 to 512×512.

Figure 7 in the main text presents the CLIP vs. LPIPS results for FlowEdit and the competing methods with both SD3 and FLUX. Below, we detail the hyperparameters used for SD3 (Tab. S1) followed by those for FLUX (Tabs. S2, S3, S4). The results shown in Figs. 6, S2 and Tabs. S5 and S6 were obtained using the parameters listed in Tabs. S1, S2 and S3. The bold entries indicate the specific settings used when multiple hyperparameter options were tested.

B.2.1. Stable Diffusion 3 hyperparameters

In Fig. 7 in the main text both FlowEdit and ODE Inversion are shown with three options for the CFG target scale, as detailed in Tab. S1, from left to right. For SDEdit the different values of $n_{\max}$ represents different strength settings ranging from 0.2 to 0.8 in intervals of 0.1. The markers in the figure indicate results with these strength values from left to right.

Table S1. **SD3 hyperparameters.**

	T steps	$n_{\max}$	CFG @ source	CFG @ target
SDEdit	50	10, 15, **20**, 25, 30, 35, 40	-	13.5
ODE Inv.	50	33	3.5	**13.5**, 16.5, , 19.5
iRFDS	official implementation and hyperparameters			
FlowEdit	50	33	3.5	**13.5**, 16.5, , 19.5

B.2.2. FLUX hyperparameters

In Fig. 7 in the main text both FlowEdit and ODE Inversion are shown with three options for the CFG target scale, as detailed in Tab. S2, from left to right.

For SDEdit the different values of $n_{\max}$ represent different strength values, corresponding to $0.25,\ 0.5,\ 0.75$. The markers in the figure indicate results with these strength values from left to right.

Table S2. **FLUX hyperparameters.**

	T steps	$n_{\max}$	CFG @ source	CFG @ target
SDEdit	28	7, 14, **21**	-	5.5
ODE Inv.	28	20, **24**	1.5	3.5, 4.5, **5.5**
FlowEdit	28	24	1.5	3.5, 4.5, **5.5**

For RF-Inversion, we explore multiple sets of hyperparmeters, as the paper does not report specific ones for general editing. Following the SM of their work, we experimented with several combinations, detailed in Tab. S3 using their notations.

Table S3. **RF-Inv. hyperparameters.**

T steps	s starting time	τ stopping time	η strength
28	0	8, 7, **6**	**0.9**, 1.0

For RF Edit, we explore multiple injection scales as the paper does not report the injection scale used for image editing. All hyperparameters used for RF Edit are listed in Tab. S4 using their notations.

Lastly, as can be seen in Fig. 7, ODE Inversion on FLUX with these hyperparameters achieves a high CLIP score at the cost of a high LPIPS score, indicating it does not balance these metrics effectively. By varying $n_{\max}$, ODE Inversion could achieve lower (better) LPIPS scores at the cost of lower (worse) CLIP scores. Figure S3 illustrates the CLIP and LPIPS scores for the different methods using FLUX. Specifically, ODE Inversion is also shown with $n_{\max} = 20$ in addition to $n_{\max} = 24$ as

Table S4. **RF Edit. hyperparameters.**

Steps	Guidance	Injection
30	2	**2**, 3, 4, 5

shown in the main text. However, with these adjusted hyperparameters, ODE Inversion struggles to adhere to text prompts. RF-Inversion results with $\eta = 1.0$ are also illustrated in Fig. S3 and Tab. S6. Again, with this hyperparameter this method struggles to adhere to text prompts.

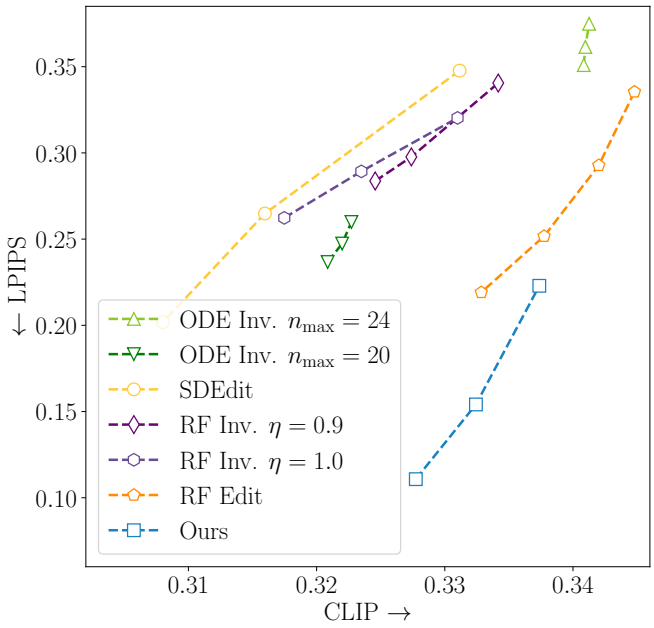

Figure S3. **Additional quantitative comparisons.** In addition to the results shown in Fig. 7 for FLUX, we include additional hyperparameters: ODE Inversion with $n_{\max} = 20$ and RF-Inversion with $\eta = 1.0$. These configurations also struggle to achieve a good balance between the CLIP and LPIPS metrics. In contrast, FlowEdit demonstrates the best balance.

B.3. Metrics comparisons

In addition to quantifying the structure preservation using LPIPS, as described in the main text, we now report alternative metrics. Specifically, we use DreamSim [5] as well as cosine similarity in the CLIP [9] and DINO [2] embedding spaces, between the source and target images' embeddings. For DreamSim, a lower score indicates better structure preservation. For CLIP images and DINO, a higher score (bounded by 1) means higher structure preservation.

Tables S5, S6 illustrate the results of these metrics, alongside LPIPS and CLIP for the hyperparameters described above. As can be seen, FlowEdit is the only method that is able to both adhere to the text prompt and preserve the structure of the source image.

Table S5. **SD3 metrics.** The first, second and third best scores are highlighted for each metric. CLIP-T measures adherence to text, while the other four scores measure structure preservation.

	CLIP-T ↑	CLIP-I ↑	LPIPS ↓	DINO ↑	DreamSim ↓
SDEdit 0.2	0.33	0.885	0.251	0.634	0.213
SDEdit 0.4	0.34	0.854	0.316	0.564	0.273
ODE Inv.	0.337	0.813	0.318	0.549	0.326
iRFDS	0.335	0.822	0.376	0.534	0.327
FlowEdit	0.344	0.872	0.181	0.719	0.253

Table S6. **FLUX metrics.** The first, second and third best scores are highlighted for each metric. CLIP-T measures adherence to text, while the other four scores measure structure preservation.

	CLIP-T ↑	CLIP-I ↑	LPIPS ↓	DINO ↑	DreamSim ↓
SDEdit 0.5	0.316	0.902	0.264	0.637	0.18
SDEdit 0.75	0.331	0.862	0.348	0.557	0.26
ODE Inv.	0.341	0.822	0.374	0.505	0.328
RF Inv.	0.334	0.856	0.34	0.558	0.266
RF Edit 2	0.344	0.833	0.335	0.53	0.32
RF Edit 5	0.332	0.876	0.22	0.65	0.22
FlowEdit	0.337	0.875	0.223	0.682	0.252

C. Effect of $n_{\max}$

As described in the main text, we define an integer to determine the starting timestep of the process, where $0 \leq n_{\max} \leq T$. The process is initialized with $Z^{\text{FE}}_{t_{n_{\max}}} = X^{\text{src}}$. When $n_{\max} = T$, the full edit path is traversed and the strongest edit is obtained. For $n_{\max} < T$, the first $(T - n_{\max})$ timesteps are skipped, effectively shortening the edit path. This is equivalent to inversion, where weaker edits are obtained by inverting up to timestep $n_{\max}$, and sampling from there. Figure S4 illustrates the effect of $n_{\max}$ on the results. The CFG and T used for these editing results are the same as mentioned in the main text, except for $n_{\max}$ whose value is specified in the figure.

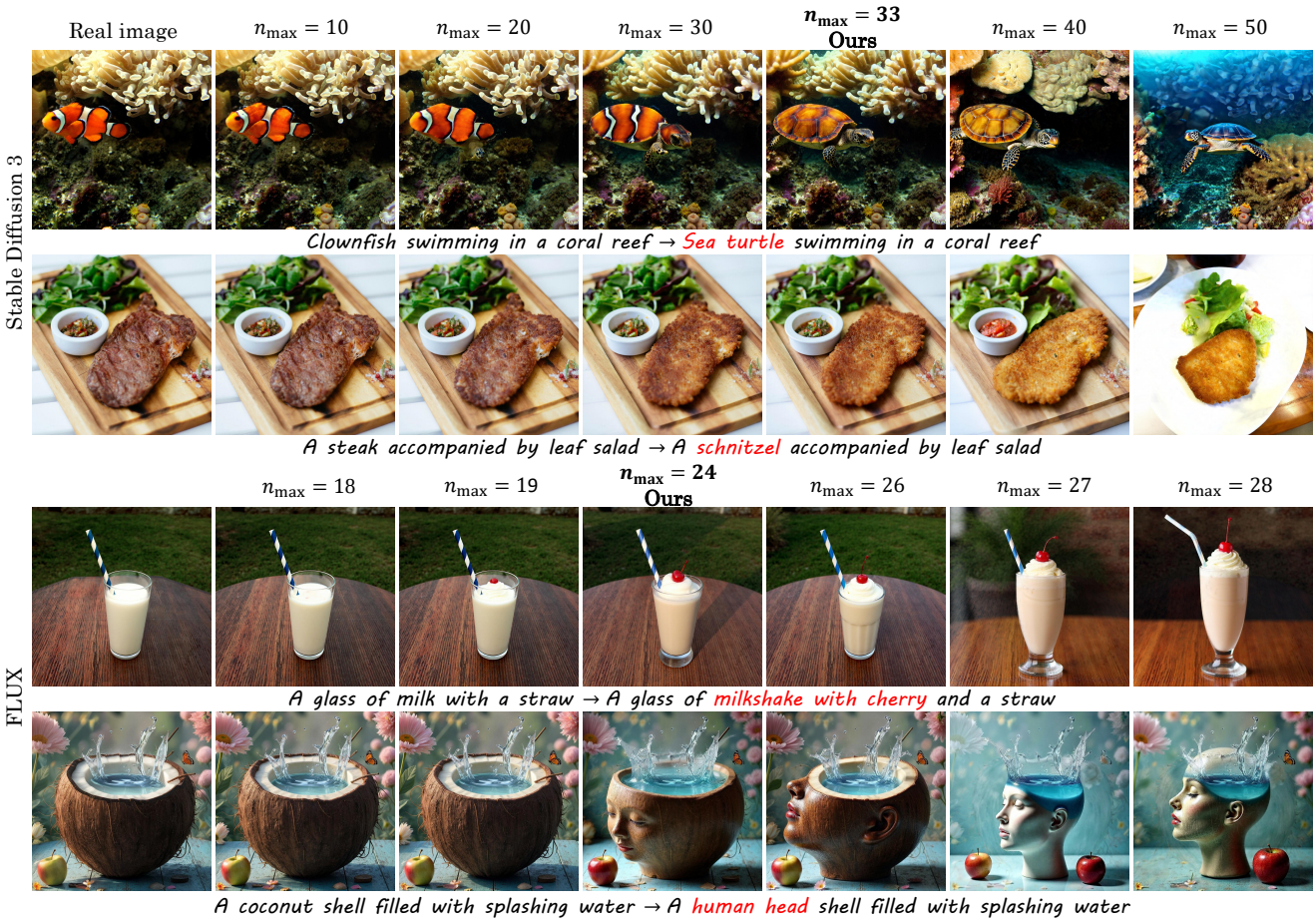

Figure S4. **Effect of $n_{\max}$.**

D. Text-based style editing

FlowEdit excels in structure preserving edits. This trait is usually desired in text-based image editing, yet in some cases it could be too limiting. One such scenario is text-based style editing, where we would like to slightly deviate from the original structure in order to achieve a stronger stylistic effect. While n_{max} allows some control over the structure, it might not be enough by itself to control the finer details required for style changes, *i.e*. higher frequency textures.

To achieve better control over the finer details, we define a new hyperparameter, n_{min}, that controls the structure deviation at lower noise levels, and effectively allows stronger modifications to the higher frequencies. Specifically, when $i < n_{\text{min}}$ we apply regular sampling with the target text, rather than following FlowEdit as described in the main text. These steps are further detailed in our full algorithm, Alg. S1.

Figure S5 illustrates the effect of n_{min} for text-based style editing. For small n_{min} the edited image preserves the structure of the original image (especially in the higher frequencies), while for higher n_{min} values it slightly deviates from it, achieving a stronger edit. These results were obtained using FLUX with $T = 28$ steps, $n_{\text{max}} = 21$ and CFG scales of 2.5 and 6.5 for the source and target conditionings, respectively. The n_{min} values are mentioned in the figure.

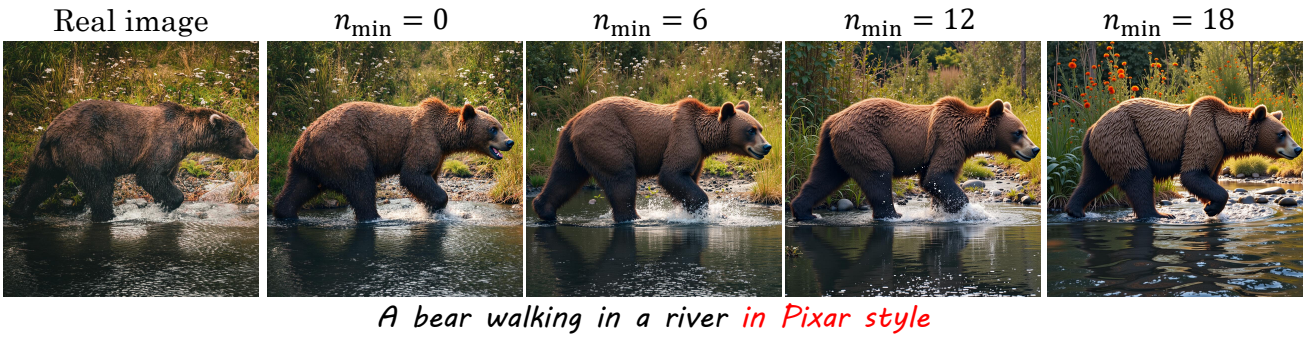

Figure S5. **Effect of n_{min}.** When n_{min} is small, the edited image remains close to the original image and struggles to align with the text. In contrast, larger values of n_{min} result in better adherence to the text but at the cost of greater deviation from the original image.

The table below includes the hyperparameters used for the text-based style editing results in Fig. 8.

Table S7. **Hyperparameters used for the results displayed in Fig. 8.**

	Model	n_{min}	n_{max}	CFG @ source	CFG @ target
House by a lake in minecraft style	SD3	15	31	3.5	13.5
Lighthouse in watercolor painting style	SD3	15	31	3.5	13.5
Kid running in anime style	FLUX	14	21	2.5	6.5
Dog in Disney style	FLUX	14	24	1.5	4.5

E. Hyperparmeters used for Fig. 1

The results in Fig. 1 were achieved using the hyperparmeters described in Tab. S8 below.

Table S8. **Hyperparameters used for the results displayed in Fig. 1.**

	Model	n_{min}	n_{max}	CFG @ source	CFG @ target
this → home	FLUX	0	24	1.5	3.5
Cat → Raccoon	FLUX	0	24	1.5	4.5
LOVE → FLOW	FLUX	0	24	1.5	4.5
Bread → Bacon	FLUX	0	24	1.5	3.5
Mountain → Volcano	FLUX	0	24	1.5	5.5
Lizard → Dragon	SD3	0	33	3.5	13.5
Man jumping in Pixar style	SD3	15	21	3.5	13.5
Bread → Steak	SD3	0	33	3.5	13.5
White dog w/ cat → Dalmatian w/o cat	SD3	0	33	3.5	13.5

F. Full Algorithm

Algorithm S1 Full FlowEdit algorithm

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

G. Limitations

FlowEdit excels in structure preserving edits, and is therefore often limited in its ability to make substantial modifications to large regions of the image. This is illustrated in Fig. S6 for pose and background editing, respectively. In these cases, FlowEdit does not fully modify the image according to the target prompt and it fails at preserving the source identity. To obtain a greater deviation from the source image, it is possible to increase $n_{\min}$, as we illustrate in App. D. This is helpful for style editing, but is often still not enough for background and pose edits.

Figure S6. **Limitations.** FlowEdit is often fails at making substantial modifications to large region of the image, as in pose (left) and background (right) editing.

H. Effect of random noise on the results

FlowEdit adds random noise to the source image. If the number of samples over which we average is small, then the algorithm is effectively stochastic. Namely, it produces different editing results with different random seeds. These variations can lead to diverse edits for the same text-image pair. As shown in Fig. S7, the rocks are transformed into different bonsai trees, and the tent appears in various locations within the image.

However, these changes can also result in failure cases, as illustrated in Fig. S8. For example, when editing a white horse into a brown one, the edited results sometimes show the horse with more than four legs or suffer from other artifacts.

The editing results in both figures were obtained using SD3 and the hyperparameter mentioned in the main text.

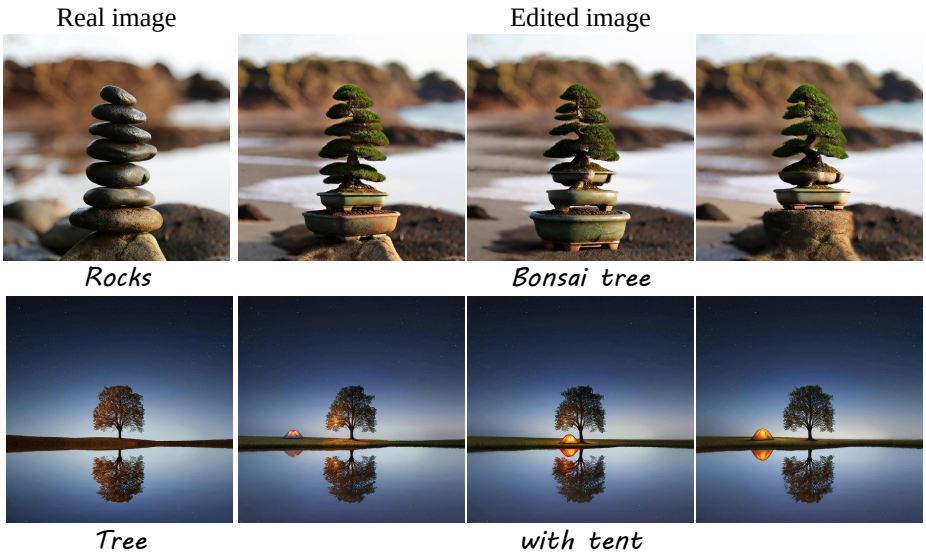

Figure S7. **FlowEdit diverse results due to different added noise.**

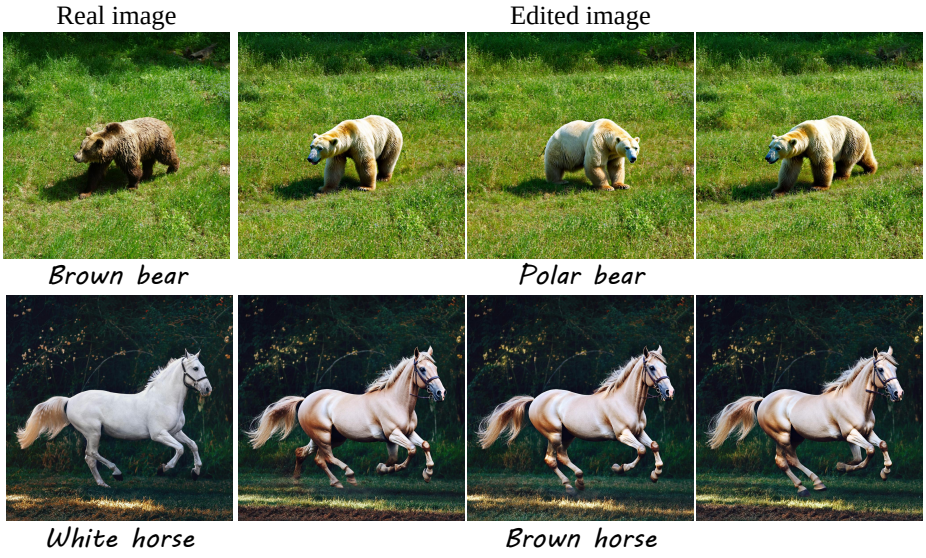

Figure S8. **Failure cases due to different added noise.**

I. Effect of source prompt on the results

In the experiments detailed in the main text we paired each image with a source prompt, generated by LLaVA-1.5 and manually refined. However, a source prompt is not required for FlowEdit and in fact has little effect on FlowEdit results, as detailed in Tab. S9. We first tested FlowEdit's sensitivity to the source prompt by generating variations on all the source prompts in the dataset using ChatGPT. We then completely omitted the source prompt and used an empty prompt. Both changes have little effect on the results. These results were obtained using SD3.

Table S9. **Effect of source prompt on the results.** CLIP-T measures adherence to text, while the other four scores measure structure preservation.

	CLIP-T ↑	CLIP-I ↑	LPIPS ↓	DINO ↑	DreamSim ↓
original source prompt	0.344	0.872	0.181	0.719	0.253
different source prompt	0.343	0.872	0.181	0.713	0.254
w/o source prompt	0.343	0.872	0.192	0.709	0.254

J. Illustration of the noise-free path between the source and target distributions

As explained in Sec. 4 in the main text, the path defined by (7) is noise-free, and it constitutes an autoregressive coarse-to-fine evolution from Z_0^{src} to Z_0^{tar}. Figure S9 illustrates this evolution for both synthetic and real images using editing by inversion (top). This path starts from the source image, $Z_1^{\text{inv}} = Z_0^{\text{src}}$. Moving along this path requires adding the vector $V_t^{\Delta}(Z_t^{\text{src}}, Z_t^{\text{tar}})$ to Z_1^{inv}. Illustrations of these vectors along the path are shown beneath the images in Fig. S9. Each $V_t^{\Delta}(Z_t^{\text{src}}, Z_t^{\text{tar}})$ image is the result of the difference between the two images above it.

It can be seen that for large t values, $V_t^{\Delta}(Z_t^{\text{src}}, Z_t^{\text{tar}})$ contains mainly low frequency components. As t decreases, higher-frequency components become increasingly visible. This stems from the fact that $V_t^{\Delta}(Z_t^{\text{src}}, Z_t^{\text{tar}})$ corresponds to the difference between $V^{\text{tar}}(Z_t^{\text{tar}}, t)$ and $V^{\text{src}}(Z_t^{\text{src}}, t)$. At large t, the noise level is substantial and therefore this vector captures mainly low-frequency components. As t decreases, the vector begins to capture higher-frequency details. This process constitutes an autoregressive coarse-to-fine evolution that starts from Z_0^{src} and ends at Z_0^{tar}, similarly to the evolution of the diffusion/flow process itself [3, 10].

Figure S9 also illustrates the evolution along the FlowEdit path and the $V_t^{\Delta}(Z_t^{\text{src}}, Z_t^{\text{tar}})$ along it. These $V_t^{\Delta}(Z_t^{\text{src}}, Z_t^{\text{tar}})$ vectors have the same characteristics as in the case of editing by inversion.

The autoregressive coarse-to-fine evolution from source to target is also schematically illustrated in Fig. S10 and empirically shown in Fig. S11. This evolution is illustrated in the frequency domain, using the power spectral density (PSD) transformation, following Rissanen et al. [10].

By arranging the images $Z_t^{\text{inv/FE}}$ along this path in a video, we can animate the interpolation between the source and target images. Additionally, by using the resulting target image as the input for subsequent editing steps, we can create a smooth animation that transitions from a source image to multiple edits. This interpolation between the source and target image can be seen in Fig. S12 for editing by inversion and in Fig. S13 for FlowEdit. This noise-free path reveals, for example, how gradually a tiger becomes a bear. Furthermore, the interpolation between a cat image and a fox image, going through lion, tiger and a bear can also be seen at the end of the supplementary video.

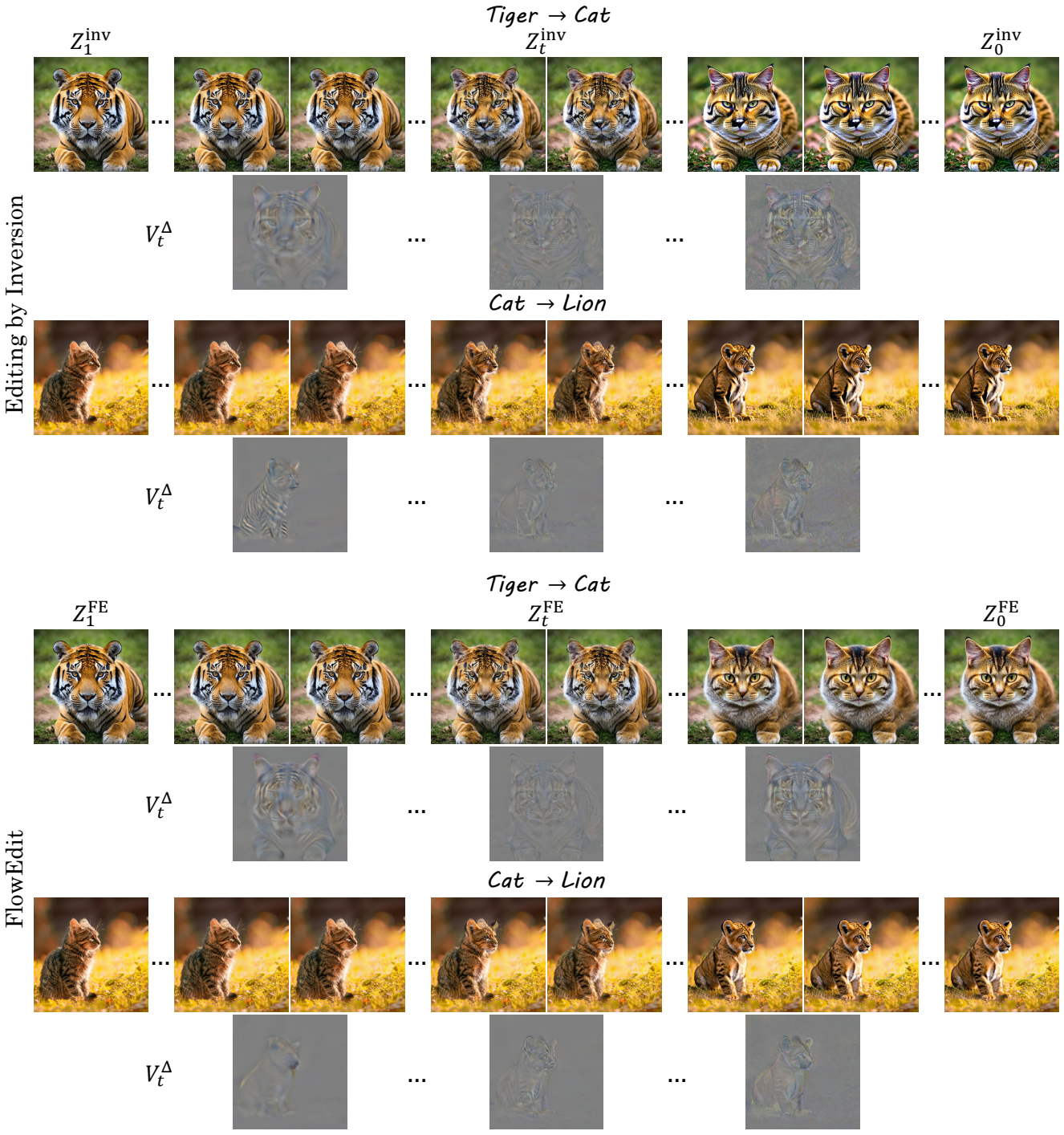

Figure S9. **Illustration of the noise free path and V^{Δ}.**

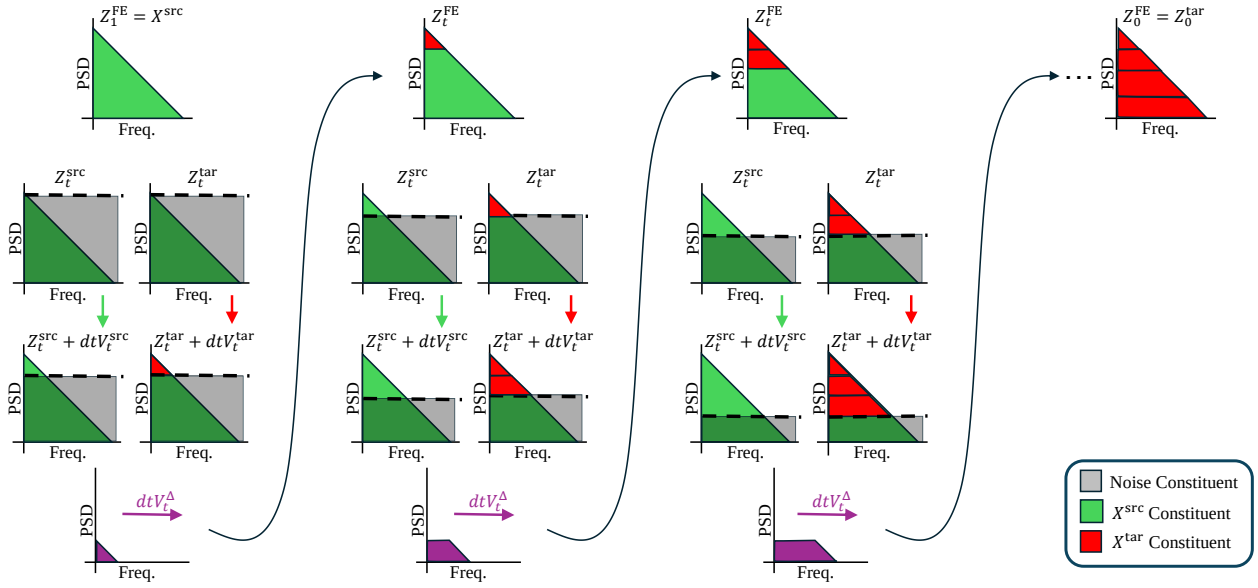

Figure S10. **Schematic illustration of the spectral behavior of FlowEdit's marginals.** Green and red markings on the graph depict the source and target image's spectral constituents, respectively. The gray markings depict the added white Gaussian noise. The top row illustrates our direct ODE path (9). The second row illustrates our noisy marginals (Sec. 5). The third row illustrates the result of a flow step with step-size dt towards the clean distributions. The fourth row illustrates dtV_t^{Δ}. The vector dtV_t^{Δ} captures the differences between the new source and target frequencies that were "revealed" during the denoising step. This dtV_t^{Δ} is used to drive our ODE process to the next timestep. This is in line with the noise-free, coarse-to-fine behavior we observed in our experiments, and as seen with the image examples in Fig. S9. We also provide an empirical evaluation for the illustration in S11.

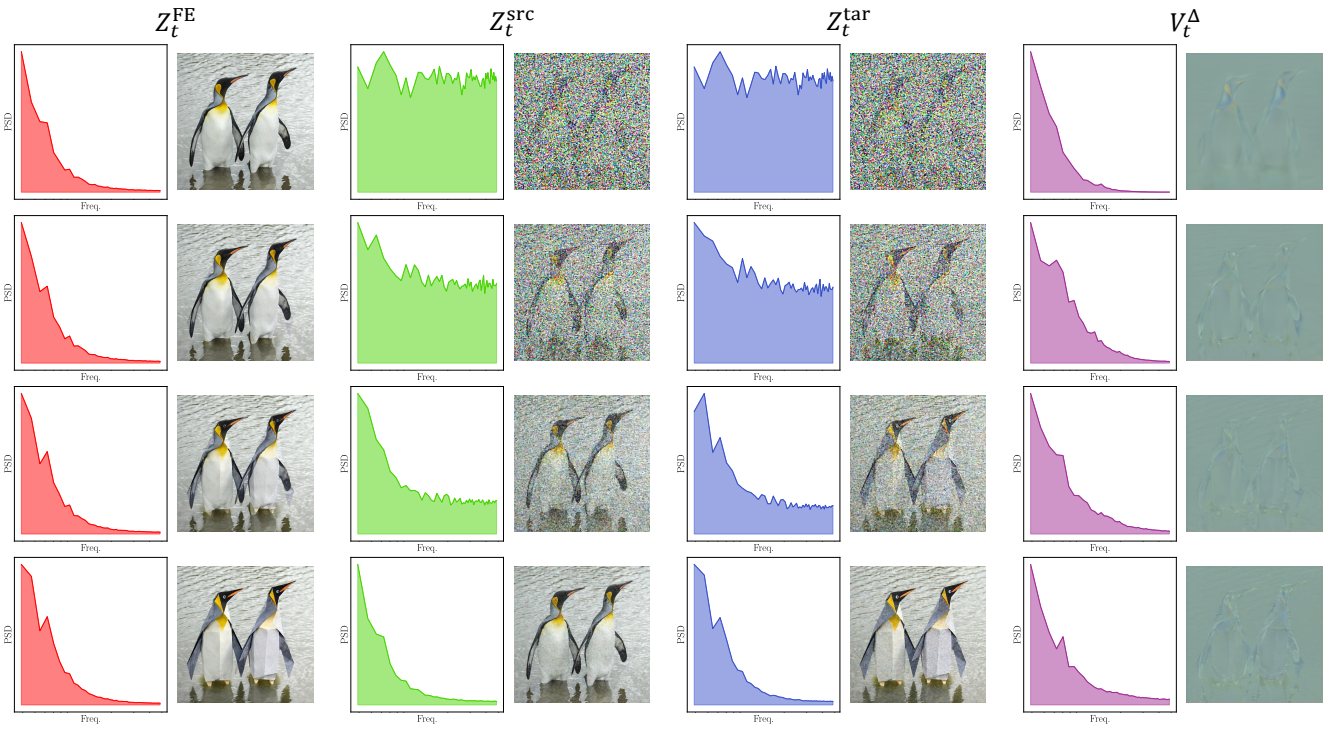

Figure S11. **Empirical evaluation of the spectral behavior of FlowEdit's marginals.** The PSD transform [10] is shown for example images along the FlowEdit path (left), the corresponding noisy versions of the source and target images (middle), and the associated velocity fields (right). The PSDs were averaged over four different edits. The example images are the results of one of these edits, corresponding to penguins $\rightarrow$ origami penguins. The first column pair shows the PSD and images of our ODE path Z_t^{FE} (9), progressing from top to bottom. It can be seen that the spectra of these clean images is nearly the same across all timesteps. The second and third column pairs show the noisy marginals, and as can be seen both $\hat{Z}_t^{\text{src}}$ and $\hat{Z}_t^{\text{tar}}$ are masked with the same amount of noise, and are valid inputs to the trained flow models. Finally, the last column pair represents V_t^{Δ}, which is noise free, and holds more low-frequency edit information in the starting timesteps, and higher frequencies at the later ones. Importantly, these statistics hold for both (reinterpreted) editing-by-inversion and our method.

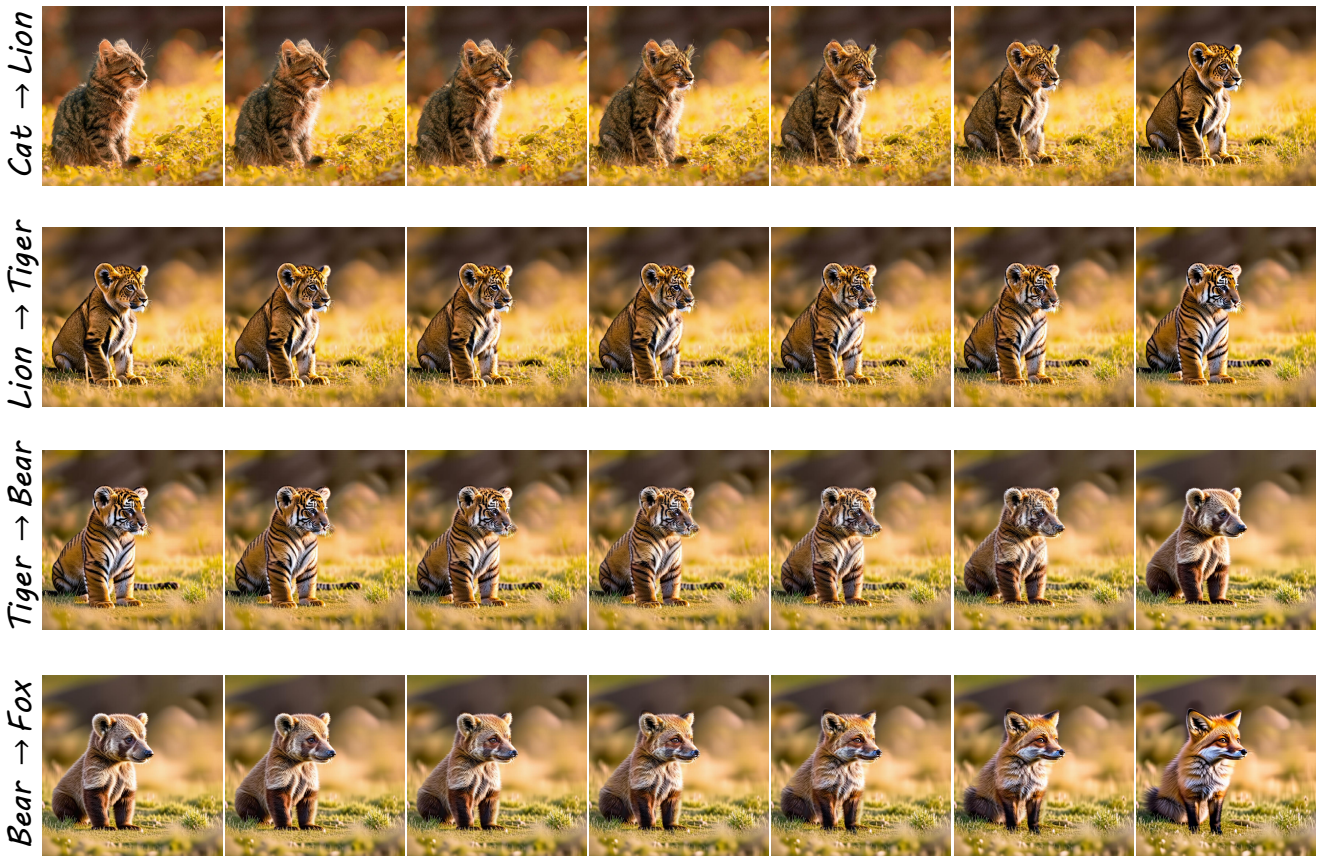

Figure S12. **Results of editing by inversion along the noise-free path.** The cat image on the top left is used as the input to editing by inversion, where the target prompt is a lion. Then, the lion image is used as input and so forth. It can be seen that the edited images do not fully preserve the structure and fine details of the original image, *e.g* the grass around the cat.

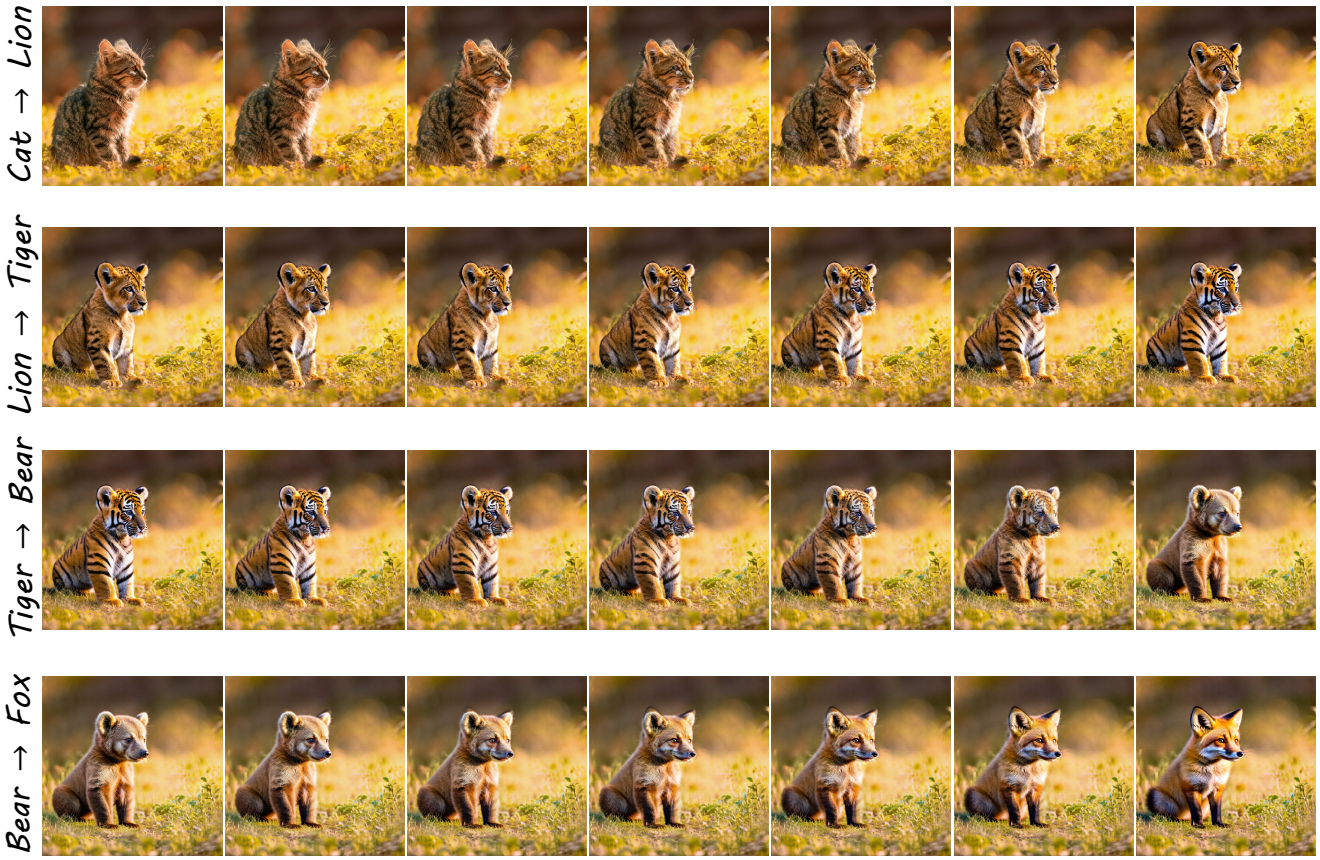

Figure S13. **Editing results of FlowEdit along the noise free path.** The cat image on the top left is used as the input to FlowEdit, where the target prompt is a lion. Then, the lion image is used as input and so forth. It can be seen that the edited images preserve the structure and the fine details of the original image, *e.g* the grass around the cat, even after multiple edits.

K. Effect of n_{avg}

FlowEdit operates by evaluating $V_t^{\Delta}(\cdot)$ on multiple realizations of $\hat{Z}_t^{\text{src}}$ and $\hat{Z}_t^{\text{tar}}$ at each t. Then, this $V_t^{\Delta}(\cdot)$ is used to drive our ODE (9). This becomes impractical in cases where model evaluations are expensive (SD3/FLUX). Fortunately, averaging also occurs across timesteps when the noises $\{N_t\}$ are chosen to be independent across timesteps. This means that with large enough T, we can use a smaller n_{avg}, reducing expensive model evaluations. Specifically, the default values of T for SD3 and FLUX are high enough for our method to perform well with $n_{\text{avg}} = 1$, relying purely on averaging across t.

Figure S14 illustrates this effect. As the value of n_{avg} increases, the LPIPS distance decreases. We used SD3 and n_{avg} values of $1,\ 3,\ 5,\ 10$. For a large number of discretization and editing steps, *i.e.* $T = 50,\ n_{\text{max}} = 33$ (orange curve), the number of averaging steps has little effect on the results, as averaging already occurs between timesteps. Hence, increasing n_{avg} only slightly improves the LPIPS score and has a negligible effect on the CLIP score (note that the horizontal axis ticks spacing is 0.001). These hyperparameters, $T = 50,\ n_{\text{max}} = 33$, with $n_{\text{avg}} = 1$, are the hyperparameters used in our method. However, for a small number of discretization and editing steps, *i.e.* $T = 10,\ n_{\text{max}} = 7$ (purple curve), increasing n_{avg} has a substantial effect. It reduces the LPIPS distance by ~ 0.3 and increases the CLIP score by 0.035. In both experiments, the CFG scales are the same as those described in the main text.

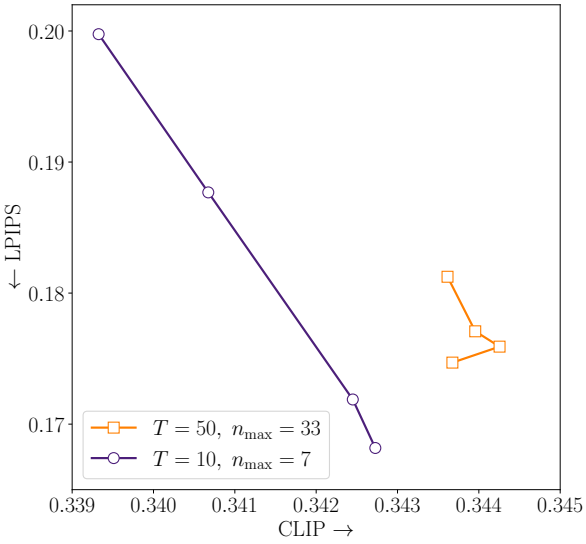

Figure S14. **CLIP vs. LPIPS for different values of n_{avg}.** From top to bottom, the markers correspond to n_{avg} of $1,\ 3,\ 5,\ 10$.

L. Further discussion on the relation to optimization based methods

As detailed in the main text, it is unnatural to view FlowEdit as an optimization based method, like DDS or PDS, because (i) it does not choose timesteps t at random but rather following a monotonically decreasing schedule $\{t_i\}_{i=0}^{n_{\max}}$, and (ii) it must use a learning rate of exactly $dt = t_{i-1} - t_i$ at iteration i. In Sec. L.3 below, we show that even a slight change of the step size in FlowEdit leads to rapid deterioration of output quality. This suggests that it is more natural to view FlowEdit as solving an ODE rather than an optimization problem.

Point (i) above is a major distinction between FlowEdit and DDS/DPS, as it guarantees that the model is always fed with inputs from the distribution on which it has been trained. When sampling timesteps at random, as in DDS and DPS, there is a possibility of drawing a small timestep t at the beginning of the process, when the image has not yet been modified. This leads to out-of-distribution inputs to the model. For example, in the cat→lion edit of Fig. S13, if DDS draws a small t at the beginning of the process, then the model is fed with a cat image that is only slightly noisy (where the cat is clearly visible), and is tasked with denoising it conditioned on the text "lion". Clearly, a slightly noisy cat image is out-of-distribution under the condition "lion".

However, it turns out that there is an even more fundamental issue with the optimization viewpoint. Specifically, we claim that the optimization viewpoint is not fully justified even for DDS, as the loss that it attempts to minimize does not actually decrease throughout the DDS iterations. We thoroughly evaluate this in the next subsection.

L.1. Delta denoising loss

The DDS method was presented as an iterative approach for minimizing the delta denoising (DD) loss. The DD loss for matched and unmatched image-text embedding pairs $\mathbf{z}, y, \hat{\mathbf{z}}, \hat{y}$ respectively, is defined as

$$\mathcal{L}_{\text{DD}}(\phi, \mathbf{z}, y, \hat{\mathbf{z}}, \hat{y}, \epsilon) = \int_0^1 \|\epsilon^\omega(\mathbf{z}_t, y, t) - \epsilon^\omega(\hat{\mathbf{z}}_t, \hat{y}, t)\|^2 dt, \tag{S1}$$

where t is the diffusion timestep and $\epsilon^\omega(\cdot)$ is the trained (noise-predicting) model with guidance scale ω [6]. The DDS iterations constitute an approximation for a stochastic gradient descent process. However, as we illustrate in Fig. S15, this approximation is quite poor. Specifically, as can be seen for two editing examples, the DDS iterations do not decrease the DD loss. They rather tend to increase it. Furthermore, as illustrated in Fig. S15, if the optimization is allowed to continue, the editing results deteriorate.

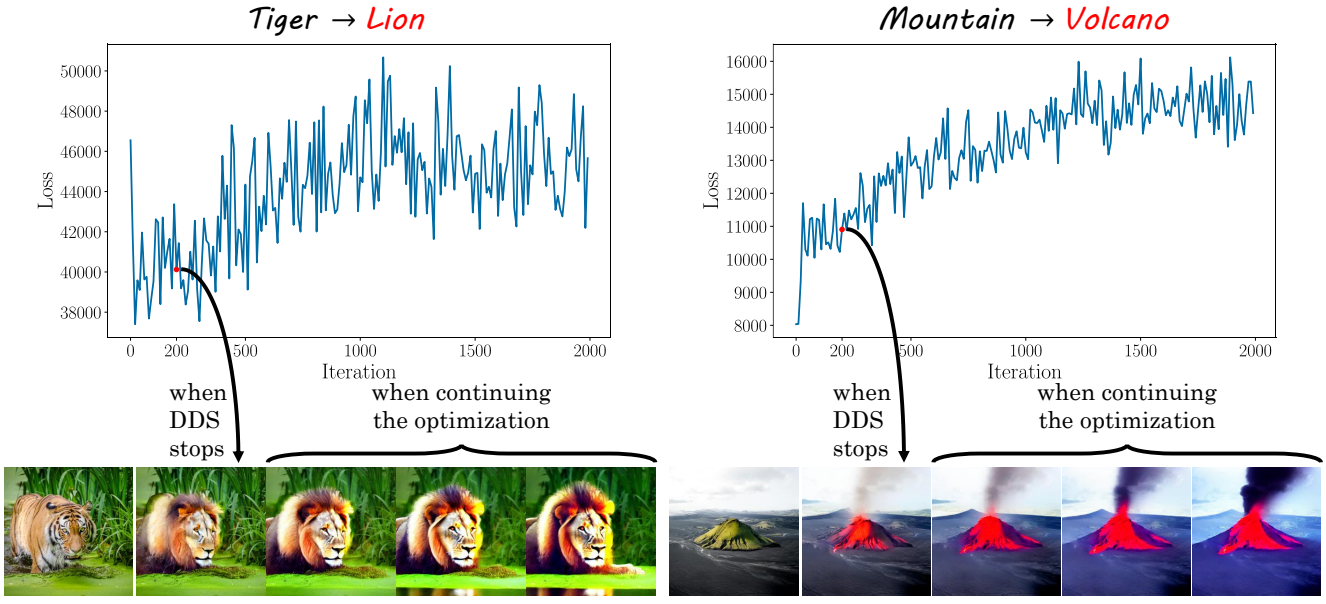

Figure S15. **DDS optimization process.**

These experiments were carried out using the official DDS implementation[1], with SD 1.5 [11]. We allowed the DDS optimization process to continue beyond the default 200 iterations used in the official implementation by running the optimization process an additional 1800 iteration, to a total of 2000 iterations. Every 100 iterations we calculated the DD loss ((S1)) for 19 timesteps between 50 and 950 with spacing of 50. We then summed the loss for all these timesteps to obtain the loss vs. iteration graphs in Fig. S15.

We do not compare FlowEdit with DDS results directly, as the comparison will not be fair due to the different backbone models, but we rather shed light on this strange behavior of the DDS optimization process.

L.2. FlowEdit with different step sizes

When using FlowEdit with a step size that is either smaller or larger than $dt = t_{i-1} - t_i$, the results deteriorate, as illustrated in Fig. S16, qualitatively and quantitatively. We evaluate FlowEdit using SD3 with step-size, dt, scaled by a factor of c. Only for $c = 1$ we get a favorable balance between the CLIP and LPIPS metrics, *i.e.* a high CLIP score and a low LPIPS score. Furthermore, we see that even slight deviations from $c = 1$ lead to a significant drop in performance. This can also be clearly seen in the qualitative example, where the edited result adheres to the text prompt and is free of artifacts only for $c = 1$.

This behavior indicates that FlowEdit cannot be considered as a gradient descent (GD) optimization over a loss function. In GD, step sizes with similar values yield similar results, and the optimization dynamics are generally smooth, without abrupt changes. This is different from FlowEdit results illustrated in Fig. S16, where a big difference in the results occurs with c values around 1. In fact, as FlowEdit solves an ODE that traverses a path between the source distribution and the target distribution, the step sizes along this path need to sum to 1. Arbitrarily scaling the step-size by some constant violates this requirement and therefore leads to deteriorated results.

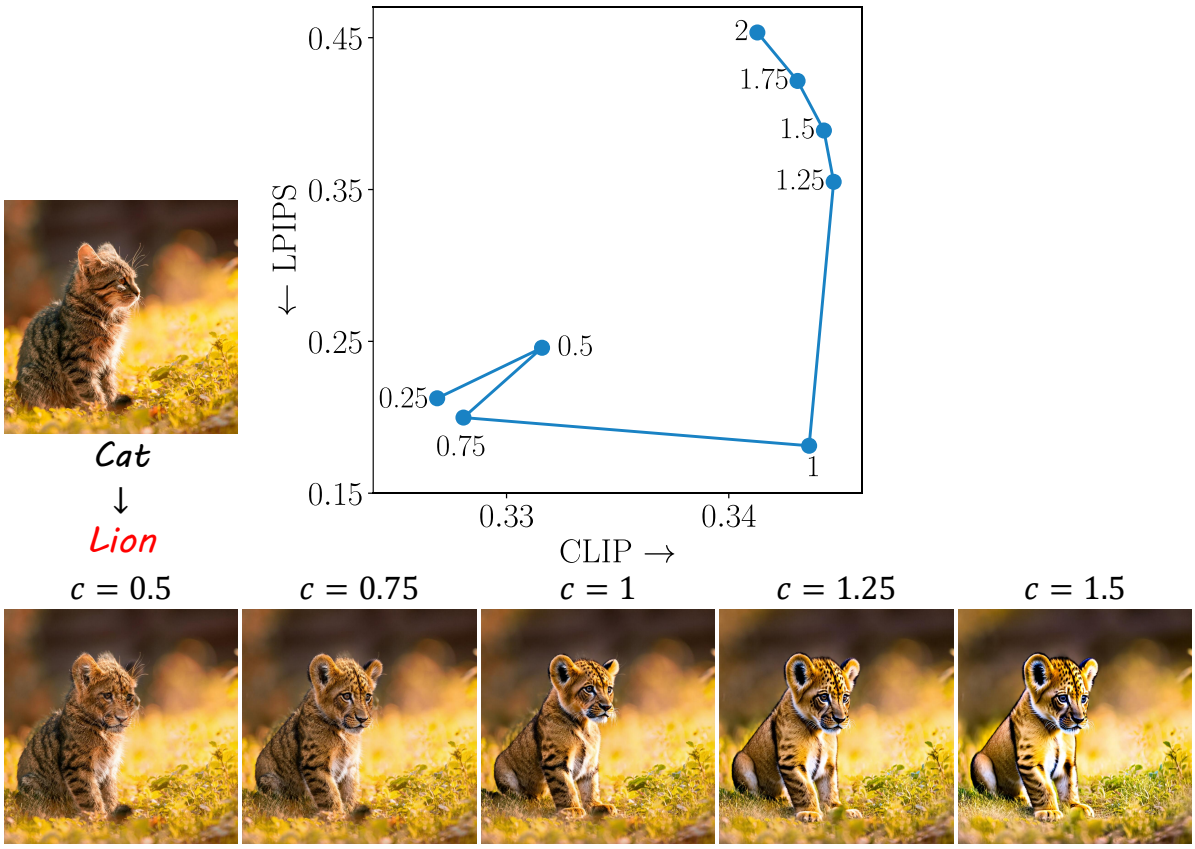

Figure S16. **FlowEdit results with a scaled step size.** The scale parameter c that multiplies the step-size is indicated next to each point.

[1] https://github.com/google/prompt-to-prompt/blob/main/DDS_zeroshot.ipynb

L.3. Mathematical discrepancies between the FlowEdit, DDS, PDS update steps

Regardless of the key differences in the ODE vs. optimization viewpoints, here we focus on the difference between the update steps themselves, translated to a common diffusion denoising formulation. To do so, we rewrite our ODE step V_t^{Δ} in terms of differences in noise predictions. Specifically, assuming $X_t = (1-t)X_0 + t\epsilon_t$, we will express the velocity field, which is given by $V(x_t, t) = \mathbb{E}[\epsilon_t - X_0 | X_t = x_t]$ (see [8]) in terms of the noise prediction $\hat{\epsilon}(x_t, t) \triangleq \mathbb{E}[\epsilon | x_t = x_t]$. We have

$$\begin{aligned} V(x_t, t) &= \mathbb{E}[\epsilon - X_0 | X_t = x_t] \\ &= \frac{1}{1-t}\mathbb{E}[(1-t)\epsilon - (1-t)X_0 | X_t = x_t] \\ &= \frac{1}{1-t}\mathbb{E}[\epsilon - ((1-t)X_0 + t\epsilon) | X_t = x_t] \\ &= \frac{1}{1-t}(\mathbb{E}[\epsilon | X_t = x_t] - x_t) \\ &= \frac{1}{1-t}(\hat{\epsilon}(x_t, t) - x_t). \end{aligned} \tag{S2}$$

Therefore,

$$\begin{aligned} V_t^{\Delta}(\hat{Z}_t^{\text{src}}, \hat{Z}_t^{\text{tar}}) &= V^{\text{tar}}(\hat{Z}_t^{\text{tar}}, t) - V^{\text{src}}(\hat{Z}_t^{\text{src}}, t) \\ &= \frac{1}{1-t}\left(\hat{\epsilon}(\hat{Z}_t^{\text{tar}}, t) - \hat{Z}_t^{\text{tar}}\right) - \frac{1}{1-t}\left(\hat{\epsilon}(\hat{Z}_t^{\text{src}}, t) - \hat{Z}_t^{\text{src}}\right) \\ &= \frac{1}{1-t}\left(\left(\hat{\epsilon}(\hat{Z}_t^{\text{tar}}, t) - \hat{\epsilon}(\hat{Z}_t^{\text{src}}, t)\right) - \left(\hat{Z}_t^{\text{tar}} - \hat{Z}_t^{\text{src}}\right)\right) \\ &= \frac{1}{1-t}\left(\left(\hat{\epsilon}(\hat{Z}_t^{\text{tar}}, t) - \hat{\epsilon}(\hat{Z}_t^{\text{src}}, t)\right) - \left(Z_t^{\text{FE}} - tX_0^{\text{src}} + t\epsilon_t - (1-t)X_0^{\text{src}} - t\epsilon_t\right)\right) \\ &= \frac{1}{1-t}\left(\left(\hat{\epsilon}(\hat{Z}_t^{\text{tar}}, t) - \hat{\epsilon}(\hat{Z}_t^{\text{src}}, t)\right) - \left(Z_t^{\text{FE}} - X_0^{\text{src}}\right)\right), \end{aligned} \tag{S3}$$

where the fourth transition comes from our construction of $\hat{Z}_t^{\text{src}} = (1-t)Z_0^{\text{src}} + t\epsilon_t$ and $\hat{Z}_t^{\text{tar}} = Z_t^{\text{FE}} + \hat{Z}_t^{\text{src}} - Z_0^{\text{src}}$ which we discussed in the main text.

As we discussed earlier, the update step in DDS is $\eta(t)(\hat{\epsilon}(Z_t^{\text{tar}}, t) - \hat{\epsilon}(Z_t^{\text{src}}, t)$, and is different from our update step, since it only relies on the differences between the noise predictions.

Regarding PDS[7], the proposed update step in Eq. (14) in their paper is of the form

$$\nabla \mathcal{L}_{\text{PDS}} = \psi(t)(Z_t^{\text{PDS}} - X_0^{\text{src}}) + \chi(t)(\hat{\epsilon}(Z_t^{\text{tar}}, t) - \hat{\epsilon}(Z_t^{\text{src}}, t), \tag{S4}$$

where Z_t^{PDS} is the variable that is being optimized, and $\psi(t), \chi(t)$ are diffusion-dependent coefficients. Note that their update step can only coincide with ours (up to a scalar multiplier) if $\forall t: \ \psi(t) = -\chi(t)$, which is not the case in their paper. Moreover, the authors of PDS state that their optimization scheme works when t is selected at random, and if chosen sequentially (mimicking FlowEdit) the gradients zero out (see the paragraph after Eq. (29) in PSD supplementary material). This further exacerbates the difference between the methods.

We conclude that even though DDS and PDS appear similar, when reformulating our update step in diffusion terms, the mathematical differences become clearer. This is added to the fact that both DDS and PDS attempt to solve an optimization problem (minimize differences between two vectors), which as we showed earlier is not fully justified.

M. Additional details about the Cats-Dogs experiment

In Sec. 5.3 we described experiments evaluating the reduced transport cost of FlowEdit compared to editing by inversion starting with generating a synthetic dataset of cat images. To generate the 1000 cat images we used variations of the prompt "a photo of cat" generated by Llama3 [4] 8B instruction model. Specifically, we requested 1000 source prompts describing cat images by providing the instruction: "Output 1000 possible prompts to a text-to-image model. These prompts should be a variation of 'a photo of a cat.', by adding descriptions and adjectives." We used these 1000 prompts as input to SD3 and generated with them 1000 cats images. Examples of these generations can be seen in the upper part of Fig. S17, covering the blue shape.

To create corresponding edited dog images, we applied both editing by inversion and FlowEdit using target prompts identical to the source prompt, except for replacing the word "cat" with "dog". Examples of these editing results can be seen in the middle of Fig. S17, covering the orange shape for FlowEdit results and covering the yellow shape for editing by inversion. Specifically, the four dog images in the middle of each shape are the results of editing the four cat images in the middle of the blue shape. It can be seen that FlowEdit editing are better when compare to editing by inversion results. We also calculate the transport cost between the cats distribution and both dogs distributions. We did it by calculating the average squared distance between SD3 latent representation of the cat images and their paired dog images, for both edits. We also calculated LPIPS between the original cat images and their paired dog images. As presented in Sec 5.3, in both metrics, FlowEdit achieved lower transport cost, compared to editing by inversion.

In addition, we generated 1000 dog images using SD3 and the same text prompts used to generate the cat images, but with "cat" replaced by "dogs". Examples of these generations can be seen in the bottom part of Fig. S17, covering the green shape. To asses the alignment between the edited dogs distribution and the generated dogs distributions we used FID and KID scores. As shown in Sec 5.3, FlowEdit achieved lower FID and lower KID scores, indicating our ability to produce images from the target distribution.

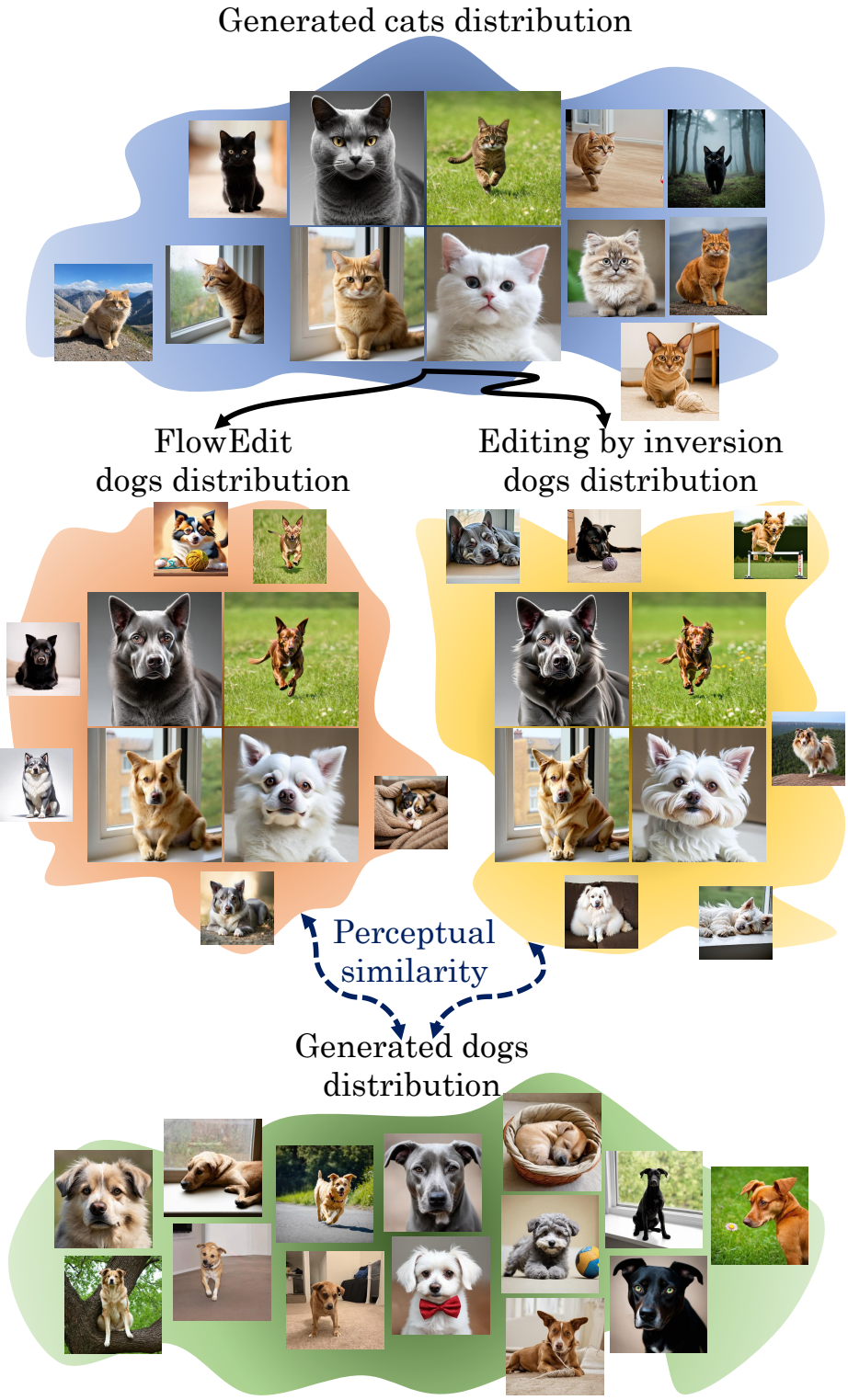

Figure S17. **Illustration of the Cats-Dogs experiment.**

References

[1] Omri Avrahami, Or Patashnik, Ohad Fried, Egor Nemchinov, Kfir Aberman, Dani Lischinski, and Daniel Cohen-Or. Stable flow: Vital layers for training-free image editing. *arXiv preprint arXiv:2411.14430*, 2024.

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[3] Sander Dieleman. Diffusion is spectral autoregression, 2024.

[4] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[5] Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. *Advances in Neural Information Processing Systems*, 36, 2024.

[6] Amir Hertz, Kfir Aberman, and Daniel Cohen-Or. Delta denoising score. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2328–2337, 2023.

[7] Juil Koo, Chanho Park, and Minhyuk Sung. Posterior distillation sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13352–13361, 2024.

[8] Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

[9] 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.

[10] Severi Rissanen, Markus Heinonen, and Arno Solin. Generative modelling with inverse heat dissipation. In *The Eleventh International Conference on Learning Representations*, 2023.

[11] 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.