SOURCESHIELD: SOURCE-BRANCH ADVERSARIAL DISRUPTION FOR INVERSION-FREE IMAGE EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-guided image editing enables controllable visual content generation, but also introduces potential risks of misuse. Adversarial disruptions can mitigate such risks by slightly perturbing the input image. However, existing disruption methods have been limited to defending against training-based or inversion-based image editing, leaving the more efficient, inversion-free paradigm unexplored. In this paper, we identify that existing methods struggle to generalize to this paradigm because they overlook the role of the editing trajectory. As a result, early-stage disruptions may be attenuated by later solver updates, and output-stage disruptions can only capture the final disruption effect. To address these limitations, we propose SourceShield, the first disruption method against inversion-free image editing that disrupts the editing trajectory in the source branch. In particular, SourceShield does not choose the target branch due to its target-prompt-specific nature and slow optimization. Specifically, SourceShield disrupts the sum of source-conditioned velocity terms across solver timesteps, to maximize the velocity differences between source and target branches. Experimental results demonstrate that SourceShield largely improves existing disruptions under unseen prompts, with 41.5% higher FID and 115.1% lower ImageReward. Additional results show its superior transferability to unseen models, resilience to countermeasures, and imperceptibility of perturbations.

1 INTRODUCTION

Text-guided image editing Brooks et al. (2023); Tumanyan et al. (2023); Xu et al. (2023b) facilitates controllable visual content creation, enabling users to modify an input image with natural-language prompts while preserving prompt-irrelevant content. However, this flexibility also introduces a practical security risk: once an image is released online, a malicious editor can use arbitrary prompts as guidance to edit the image into deceptive variants at scale. Adversarial disruptions (Song et al., 2025; Salman et al., 2023; Ozden et al., 2026) have been extensively studied to mitigate this risk by adding an imperceptible perturbation to the input image before release, producing a protected image that remains visually similar to the original but degrades potential editing of it.

Existing studies have been focused on disruptions against training-based (Brooks et al., 2023; Wu et al., 2025; Ruiz et al., 2023; Ye et al., 2023) and inversion-based image editing (Tumanyan et al., 2023; Cao et al., 2023). However, none of them has attempted to disrupt the more efficient, inversion-free image editing, which requires neither parameter adaptation nor explicit inversion, but directly updates edited variables through velocity differences between an asymmetric target branch and source branch (Xu et al., 2023b; Kulikov et al., 2025).[1]

In this paper, we address this research gap by exploring adversarial disruptions against inversion-free image editing. First, we identify the poor generalization of existing disruptions to inversion-free image editing and attribute it to their overlooking of the editing trajectory: (1) Existing early-stage disruptions (Song et al., 2025; Chen et al., 2024; Li et al., 2024; Choi et al., 2025; Son et al., 2024), which perturb intermediate features into early solver states, cannot yield persistent disruptions because their effects may be attenuated by subsequent solver updates. (2) Existing output-stage disruptions (Xue et al., 2024; Salman et al., 2023; Ozden et al., 2026; Hong et al., 2025), which optimize

[1] Detailed related work is provided in Appendix A.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

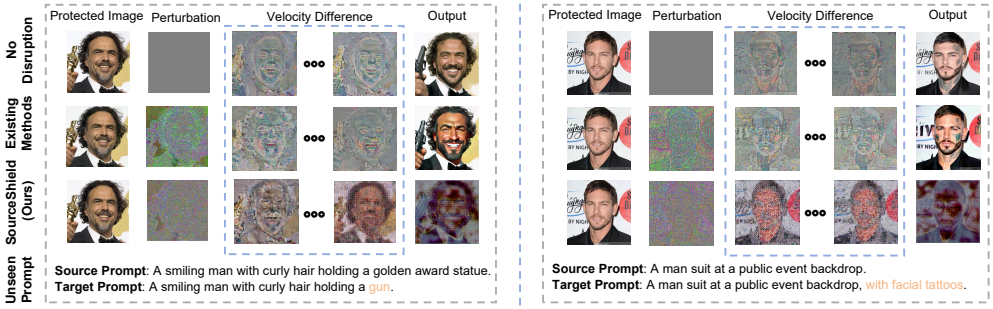


Figure 1: By considering disruptions over the editing trajectory, our SourceShield yields strong effects across internal timesteps and finally the output image. shifts the editing trajectory by persistently perturbing velocity differences, leading to degraded edited outputs.

the final edited output, are in principle not able to model the accumulated disruption effects across solver timesteps. Our analyses, through decomposing the editing trajectory induced by the differential update process of inversion-free editing, empirically support our observation by showing the small velocity differences between the source and target branches induced by existing disruptions (See Sec. 3.2). To address the above limitations of existing work, we propose SourceShield, the first disruption method against inversion-free image editing that disrupts the editing trajectory in the source branch. Specifically, SourceShield disrupts the sum of source-conditioned velocity terms across solver timesteps, to maximize the velocity differences between source and target branches. In this way, as illustrated in Fig. 1, SourceShield yields strong disruptions across internal timesteps and finally the output image.

In particular, SourceShield chooses the trajectory in the source branch rather than the target branch for two reasons. First, the target branch is target-prompt-specific: it operates on a specific target prompt, which is controlled by the malicious editor and not accessible by the image protector. In contrast, the source branch is target-prompt-agnostic, making it suitable for disruptions under unseen target prompts. Second, the target branch is temporally coupled: it requires computing gradients through the time-ordered trajectory of edited variables, increasing computational costs for optimization. In contrast, the source branch can compute gradients independently per timestep, resulting in our memory-efficient design via a two-stage chunk-wise gradient optimization, which first computes the source direction and then accumulates vector-Jacobian products over chunks of solver timesteps. Our contributions are summarized as follows:

- We, for the first time, study adversarial disruptions against the new paradigm of inversion-free image editing. In particular, we attribute the insufficiency of existing methods, which focus on disrupting training-based or inversion-based image editing, to their overlooking of the editing trajectory.
- We propose SourceShield, which disrupts the editing trajectory in the source branch, due to its target-prompt-agnostic nature and decomposability for efficient optimization. Specifically, it disrupts source-conditioned accumulation across solver timesteps to maximize the source-target velocity differences.
- We demonstrate that SourceShield outperforms existing methods across five metrics under unseen prompts, e.g., 41.5% higher FID and 115.1% lower ImageReward. It is also more transferable to unseen editing models and more resilient to countermeasures, as well as yielding more imperceptible perturbations.

2 Preliminary

2.1 Inversion-Free Image Editing

Inversion-free image editing (Kulikov et al., 2025; Xu et al., 2023b) edits an input image x^{in} from a source prompt φ^{src} to a target prompt φ^{tgt}, producing an edited output $x_{t_0}^{\text{edit}}$. It directly constructs source variables x_t^{src}, target variables x_t^{tgt}, and edited variables x_t^{edit}, and updates the edited vari-

ables through velocity differences. Following Rectified Flow (RF) (Liu et al., 2022), we denote the prompt-conditioned vector field of the pretrained generative backbone by $v_\theta(x_t, t, \varphi)$, where x_t is a trajectory variable and $t \in [0, 1]$ indexes the interpolation between data and noise endpoints. Under this formulation, the source variable at time t admits the closed-form interpolation:

$$x_t^{\mathrm{src}} = (1-t)x^{\mathrm{in}} + t\xi_t, \qquad \xi_t \sim \mathcal{N}(0, I). \tag{1}$$

The target variable is constructed from the current edited variable as

$$x_t^{\mathrm{tgt}} = x_t^{\mathrm{edit}} + x_t^{\mathrm{src}} - x^{\mathrm{in}}. \tag{2}$$

The edit direction is the difference between target- and source-conditioned velocities:

$$v_\theta^{\Delta}\left(x_t^{\mathrm{src}}, x_t^{\mathrm{tgt}}\right) = v_\theta^{\mathrm{tgt}}\left(x_t^{\mathrm{tgt}}, t, \varphi^{\mathrm{tgt}}\right) - v_\theta^{\mathrm{src}}\left(x_t^{\mathrm{src}}, t, \varphi^{\mathrm{src}}\right). \tag{3}$$

We call the two terms the target-conditioned velocity and source-conditioned velocity, defining the target branch and source branch, respectively. The edited variables are initialized at the largest solver timestep as $x_{t_T}^{\mathrm{edit}} = x^{\mathrm{in}}$. With an Euler solver, consecutive timesteps are updated by

$$x_{t_{i-1}}^{\mathrm{edit}} = x_{t_i}^{\mathrm{edit}} + \Delta t_i \, v_\theta^{\Delta}\left(x_{t_i}^{\mathrm{src}}, x_{t_i}^{\mathrm{tgt}}\right). \tag{4}$$

Here $\Delta t_i = t_{i-1} - t_i$ is the signed Euler step. Let $\{t_i\}_{i=0}^{T}$ denote the ordered solver timesteps and define $\mathcal{T} = \{1, \ldots, T\}$. Eq. (4) is applied for $i \in \mathcal{T}$ until $t_0 = 0$, where the edited output $x_{t_0}^{\mathrm{edit}}$ is obtained. Further details of RF and inversion-free image editing are provided in Appendix B.2.

2.2 Threat Model

We adopt the commonly studied threat model in existing adversarial disruption work that involves a malicious editor and an image owner.

Attacker: Malicious editor. The attacker aims to edit an image for malicious purposes using a pretrained (inversion-free) image editing model. To perform a valid edit, the attacker prepares a source prompt that captions the input image and a target prompt that specifies the desired malicious output. The target prompt is unknown to the image protector.

Victim: Image Protector. The victim protects their images that may potentially undergo malicious editing by optimizing visually imperceptible perturbations. During optimization, the victim can access a public inversion-free image editing model and obtain a source prompt that captions the input image.

In practice, source prompts from the attacker and victim are not identical but should both remain semantically similar to the input image; otherwise, inversion-free image editing itself becomes ill-posed. Overall, the perturbation is expected to remain effective under semantically similar source prompts as well as unseen target prompts and editing models.

3 Analyzing Existing Disruption Methods

In this section, we examine why existing disruption methods are insufficient for inversion-free image editing. We decompose the trajectory of edited variables to show how velocity differences accumulate across solver timesteps, and then diagnose local disruption from this trajectory-level view.

3.1 Trajectory Decomposition

Sec. 2 shows that inversion-free image editing updates the edited variable using the difference between target-conditioned and source-conditioned velocities. For the input image x^{in}, the target branch provides the target-conditioned velocity $v_\theta^{\mathrm{tgt}}(x_{t_i}^{\mathrm{tgt}}, t_i, \varphi^{\mathrm{tgt}})$, while the source branch provides the source-conditioned velocity $v_\theta^{\mathrm{src}}(x_{t_i}^{\mathrm{src}}, t_i, \varphi^{\mathrm{src}})$. From Eq. (4), the discrete update is

$$x_{t_{i-1}}^{\mathrm{edit}} = x_{t_i}^{\mathrm{edit}} + \Delta t_i \left[v_\theta^{\mathrm{tgt}}\left(x_{t_i}^{\mathrm{tgt}}, t_i, \varphi^{\mathrm{tgt}}\right) - v_\theta^{\mathrm{src}}\left(x_{t_i}^{\mathrm{src}}, t_i, \varphi^{\mathrm{src}}\right)\right], \tag{5}$$

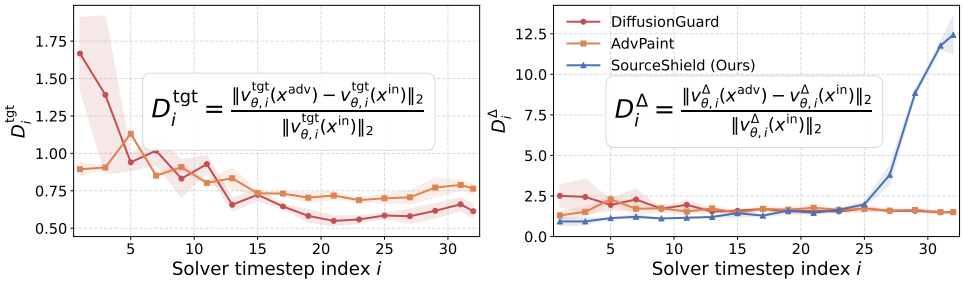


Figure 2: **Limitation illustration of existing early-stage disruptions:** Their (a) gradually decayed target-conditioned velocity deviation leads to (b) gradually decayed source-target velocity difference deviation over timesteps. In contrast, our SourceShield achieves gradually increased deviation. Here, the deviation is calculated between values for protected vs. original images.

Unrolling Eq. (5) over $i \in \mathcal{T}$ from t_T to t_0, with $x_{t_T}^{\text{edit}} = x^{\text{in}}$, gives

$$x_{t_0}^{\text{edit}}(x^{\text{in}}; \varphi^{\text{src}}, \varphi^{\text{tgt}}) = x^{\text{in}} + \underbrace{\sum_{i \in \mathcal{T}} \Delta t_i \, v_\theta^{\text{tgt}}\left(x_{t_i}^{\text{tgt}}, t_i, \varphi^{\text{tgt}}\right)}_{A^{\text{tgt}}(x^{\text{in}}; \varphi^{\text{src}}, \varphi^{\text{tgt}})} - \underbrace{\sum_{i \in \mathcal{T}} \Delta t_i \, v_\theta^{\text{src}}\left(x_{t_i}^{\text{src}}, t_i, \varphi^{\text{src}}\right)}_{A^{\text{src}}(x^{\text{in}}; \varphi^{\text{src}})}. \tag{6}$$

The underbraced terms define the accumulated contributions of the two branches: the target-conditioned accumulation $A^{\text{tgt}}(x^{\text{in}}; \varphi^{\text{src}}, \varphi^{\text{tgt}})$ from the target branch and the source-conditioned accumulation $A^{\text{src}}(x^{\text{in}}; \varphi^{\text{src}})$ from the source branch. This decomposition provides a trajectory-level view for identifying suitable disruption points.

3.2 Limitations of Existing Disruptions

We next diagnose why existing disruption strategies are insufficient for inversion-free image editing. We examine early-stage disruption with DiffusionGuard and AdvPaint, and output-stage disruption with PhotoGuard. All diagnostics use Stable Diffusion 3 (SD3).

Early-stage disruption. Early-stage disruption perturbs early signals and assumes that the disturbance will persist along the trajectory of edited variables. Fig. 2(a) shows that DiffusionGuard and AdvPaint induce early target-conditioned velocity deviations, but these deviations decrease as the solver proceeds. Consequently, after subtracting source-conditioned velocities, Fig. 2(b) shows that the velocity difference deviation D_i^{Δ} also fails to persist: DiffusionGuard decays over time, while AdvPaint exhibits only a transient early peak. This indicates that early disturbances are attenuated by later solver updates and fail to persistently affect the trajectory of edited variables. Fig. 1 shows the same trend, where early-stage disruption can still produce plausible edited outputs.

Output-stage disruption. Output-stage disruption directly degrades the final edited output and thus captures only the terminal disruption effect, without explicitly modeling how velocity differences accumulate across solver timesteps. Fig. 3 compares SourceShield with PhotoGuard in optimization time and disruption performance. SourceShield achieves stronger degradation with less optimization time, highlighting the advantage of disrupting the editing trajectory through the source branch. Therefore, due to its higher optimization cost and weaker disruption performance, we exclude output-stage disruption baselines from the subsequent main experiments. These observations further motivate trajectory-level disruption.

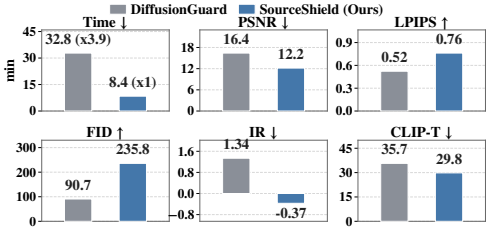


Figure 3: **Limitation illustration of output-stage disruption:** Output-stage disruption shows weaker disruption and requires longer optimization time.

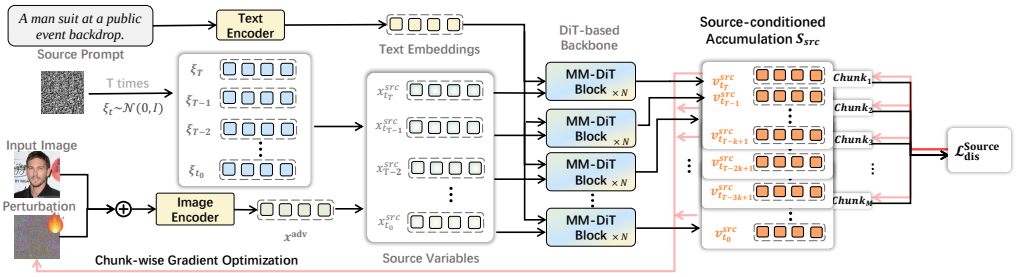


Figure 4: Overview of SourceShield. SourceShield disrupts the editing trajectory in the source branch by targeting the sum of source-conditioned velocity terms across solver timesteps, with gradients computed by two-stage chunk-wise optimization.

4 METHODOLOGY

In this section, we first formulate adversarial disruption for inversion-free image editing, then introduce SourceShield's source-branch disruption objective and two-stage chunk-wise gradient optimization. The overall pipeline of SourceShield is illustrated in Fig. 4.

4.1 PROBLEM FORMULATION

Let $x^{\text{in}} \in [0,1]^d$ denote the input image and $x^{\text{adv}} \in [0,1]^d$ denote the protected image released by the victim. The protected image is constrained within the imperceptible perturbation set $\mathcal{B}_\varepsilon(x^{\text{in}}) = \{x \in [0,1]^d : \|x - x^{\text{in}}\|_\infty \le \varepsilon\}$. We define the generic disruption objective as

$$\max_{x^{\text{adv}} \in \mathcal{B}_\varepsilon(x^{\text{in}})} \mathcal{L}_{\text{dis}}\left(x^{\text{adv}}; x^{\text{in}}\right), \tag{7}$$

where $\mathcal{L}_{\text{dis}}$ denotes the disruption objective for degrading downstream edited outputs.

4.2 SOURCE BRANCH DISRUPTION

The above analysis suggests that effective disruption should target the editing trajectory. We next explain why SourceShield disrupts the source branch instead of the target branch.

Target-conditioned accumulation. The trajectory decomposition in Eq. (6) suggests the target-conditioned accumulation A^{tgt} as a natural disruption point because it directly contributes to the edit direction. However, A^{tgt} is unsuitable: it depends on the unknown target prompt φ^{tgt} and is temporally coupled through the evolving edited variable $x_{t_i}^{\text{edit}}$. Thus, disrupting A^{tgt} requires gradients through the time-ordered trajectory of edited variables, increasing gradient-computation cost and preventing independent per-timestep computation, as derived in Appendix C.1.

Source-conditioned accumulation. In contrast, the source-conditioned accumulation A^{src} satisfies all desired properties. First, since A^{src} is accumulated across solver timesteps and subtracted from A^{tgt} in Eq. (6), disrupting it persistently perturbs the velocity differences that drive the trajectory of edited variables. Second, it is target-prompt-independent: it depends only on the source prompt φ^{src} and therefore does not require knowing the future target prompt φ^{tgt}. Third, it is timestep-wise decomposable because each source variable is constructed independently from the protected image and per-timestep noise. Consequently, its Jacobian separates over solver timesteps:

$$J_{x^{\text{adv}}} A^{\text{src}}(x^{\text{adv}}; \varphi^{\text{src}}) = \sum_{i \in \mathcal{T}} \Delta t_i \, J_{x^{\text{adv}}} v_\theta^{\text{src}}\left(x_{t_i}^{\text{src}}, t_i, \varphi^{\text{src}}\right). \tag{8}$$

This additive form enables independent per-timestep gradient computation. Therefore, we choose the source-conditioned accumulation as the disruption point.

Source-conditioned accumulation objective. SourceShield targets the source-conditioned accumulation over a selected solver timestep set $\mathcal{I} \subseteq \mathcal{T}$. For each $i \in \mathcal{I}$, it constructs the source variable from the protected image:

$$x_{t_i}^{\mathrm{src}} = (1 - t_i)x^{\mathrm{adv}} + t_i \xi_{t_i}, \qquad \xi_{t_i} \sim \mathcal{N}(0, I). \tag{9}$$

The corresponding source-conditioned accumulation is

$$S^{\mathrm{src}}(x^{\mathrm{adv}}; \varphi^{\mathrm{src}}) = \sum_{i \in \mathcal{I}} \Delta t_i \, v_\theta^{\mathrm{src}} \left(x_{t_i}^{\mathrm{src}}, t_i, \varphi^{\mathrm{src}} \right). \tag{10}$$

SourceShield maximizes its magnitude:

$$\max_{x^{\mathrm{adv}} \in \mathcal{B}_\varepsilon(x^{\mathrm{in}})} \mathcal{L}_{\mathrm{dis}}^{\mathrm{Source}}(x^{\mathrm{adv}}; \varphi^{\mathrm{src}}) = \left\| S^{\mathrm{src}}(x^{\mathrm{adv}}; \varphi^{\mathrm{src}}) \right\|_2. \tag{11}$$

Since this accumulation is subtracted in Eq. (6), amplifying it perturbs the velocity differences that drive the trajectory of edited variables. Fig. 2(b) shows that SourceShield substantially amplifies the velocity difference deviation at later solver timesteps, indicating a persistent shift in the trajectory of edited variables. We optimize Eq. (11) by projected gradient ascent:

$$x^{\mathrm{adv},r+1} = \Pi_{\mathcal{B}_\varepsilon(x^{\mathrm{in}})} \left[x^{\mathrm{adv},r} + \alpha \, \mathrm{sign} \left(\nabla_{x^{\mathrm{adv}}} \mathcal{L}_{\mathrm{dis}}^{\mathrm{Source}}(x^{\mathrm{adv},r}; \varphi^{\mathrm{src}}) \right) \right], \tag{12}$$

where α is the step size and $\Pi_{\mathcal{B}_\varepsilon(x^{\mathrm{in}})}$ projects the result back to the feasible set.

4.3 Two-Stage Chunk-Wise Gradient Optimization

Although Eq. (11) is timestep-wise decomposable, one-pass differentiation still stores activations for all selected solver timesteps, causing substantial memory consumption. To improve resource efficiency, we compute gradients with a two-stage chunk-wise vector-Jacobian product (VJP).

Stage 1: source direction. We first evaluate all selected source-conditioned velocity terms without retaining activations, accumulate S^{src}, and form the normalized source direction

$$d^{\mathrm{src}} = \frac{S^{\mathrm{src}}(x^{\mathrm{adv}}; \varphi^{\mathrm{src}})}{\| S^{\mathrm{src}}(x^{\mathrm{adv}}; \varphi^{\mathrm{src}}) \|_2}, \tag{13}$$

assuming $S^{\mathrm{src}}(x^{\mathrm{adv}}; \varphi^{\mathrm{src}}) \neq 0$. Treating d^{src} as fixed, the gradient is the sum of per-timestep VJPs:

$$\nabla_{x^{\mathrm{adv}}} \left\| S^{\mathrm{src}}(x^{\mathrm{adv}}; \varphi^{\mathrm{src}}) \right\|_2 = \sum_{i \in \mathcal{I}} \Delta t_i \left(J_{x^{\mathrm{adv}}} v_\theta^{\mathrm{src}} \left(x_{t_i}^{\mathrm{src}}, t_i, \varphi^{\mathrm{src}} \right) \right)^\top d^{\mathrm{src}}, \tag{14}$$

Stage 2: chunk-wise VJP. Let K denote the chunk size, and partition $\mathcal{I}$ into $\{\mathcal{C}_m\}_{m=1}^M$ with $|\mathcal{C}_m| \leq K$ and $M = \lceil |\mathcal{I}|/K \rceil$. For each chunk, we recompute its source-conditioned velocity terms and backpropagate the scalar projection

$$\mathcal{L}_{\mathcal{C}_m}^{\mathrm{VJP}} = \sum_{i \in \mathcal{C}_m} \Delta t_i \left\langle v_\theta^{\mathrm{src}} \left(x_{t_i}^{\mathrm{src}}, t_i, \varphi^{\mathrm{src}} \right), \mathrm{sg}(d^{\mathrm{src}}) \right\rangle, \tag{15}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient. Summing the chunk gradients recovers Eq. (14), while peak activation memory scales with K rather than the total number of selected solver timesteps $|\mathcal{I}|$.

5 Experiments

5.1 Experimental Setups

Datasets and Models. Our dataset contains 127 images from FlowEdit (Kulikov et al., 2025) and DiffVAX (Ozden et al., 2026). For each image, we generate four source-target prompt pairs: one seen pair for perturbation optimization and three unseen pairs for evaluation, yielding 127 seen and 381 unseen editing instances. We optimize perturbations on Stable Diffusion 3 (SD3) (Stability AI, 2024) and DreamShaper (DS) (Luo et al., 2023), and evaluate black-box transfer to SD3, DS, and FLUX.1-dev (FLUX) (Black Forest Labs, 2024).

Disruption baselines. We compare SourceShield with representative early-stage disruption baselines: EditShield (Chen et al., 2024), VAC-LE (Yang et al., 2025), DiffusionGuard (Choi et al., 2025), and AdvPaint (Jeon et al., 2025). More experimental details are provided in Appendix D.

Table 1: Disruption performance on unseen source-target prompts.

Methods	Unseen Prompt	Metric									
		Output Consistency						Prompt Faithfulness			
		PSNR↓		LPIPS↑		FID↑		IR↓		CLIP-T↓	
		SD3	DS	SD3	DS	SD3	DS	SD3	DS	SD3	DS
EditShield	Prompt A	14.57	19.81	0.6729	0.7818	138.2	179.7	1.00	-0.66	34.53	30.22
	Prompt B	14.46	20.08	0.6789	0.7823	119.8	184.3	0.87	-0.17	34.36	30.61
	Prompt C	14.54	19.89	0.6684	0.7818	128.8	179.2	0.96	-0.40	35.25	31.02
	Average	14.52	19.93	0.6734	**0.7820**	128.9	181.0	0.94	-0.41	34.71	30.62
VAC-LE	Prompt A	16.34	20.95	0.5119	0.4941	98.9	124.9	1.26	-0.48	35.08	31.03
	Prompt B	16.19	18.81	0.5383	0.5273	86.2	137.9	1.29	0.12	35.31	32.09
	Prompt C	16.41	20.82	0.5124	0.5028	95.5	134.4	1.18	-0.19	36.68	32.22
	Average	16.31	20.19	0.5209	0.5081	93.5	132.4	1.24	-0.18	35.69	31.78
DiffusionGuard	Prompt A	15.87	22.52	0.4712	0.4702	104.4	104.6	1.26	-0.31	34.64	30.74
	Prompt B	15.78	19.97	0.4911	0.4859	84.6	114.4	1.20	0.24	34.82	31.11
	Prompt C	15.80	22.19	0.4807	0.4734	99.4	107.8	1.16	0.07	35.83	31.96
	Average	15.82	21.56	0.4810	0.4765	96.1	108.9	1.21	-0.00	35.10	31.27
AdvPaint	Prompt A	15.10	20.63	0.6340	0.5567	122.9	146.8	1.04	-0.73	35.22	29.80
	Prompt B	14.92	18.61	0.6558	0.5775	108.0	151.3	0.76	-0.27	34.16	30.25
	Prompt C	15.16	20.27	0.6342	0.5658	114.8	152.4	0.90	-0.48	35.52	30.13
	Average	15.06	19.84	0.6413	0.5667	115.2	150.1	0.90	-0.50	34.97	30.06
SourceShield (Ours)	Prompt A	11.75	19.97	0.7735	0.6527	243.3	188.5	-0.47	-1.07	29.70	29.03
	Prompt B	11.89	18.24	0.7707	0.6674	212.5	196.6	-0.39	-0.74	29.44	28.71
	Prompt C	11.95	19.74	0.7652	0.6596	223.3	188.7	-0.54	-0.87	30.03	28.78
	Average	**11.87**	**19.32**	**0.7698**	0.6599	**226.4**	**191.2**	**-0.47**	**-0.89**	**29.72**	**28.84**

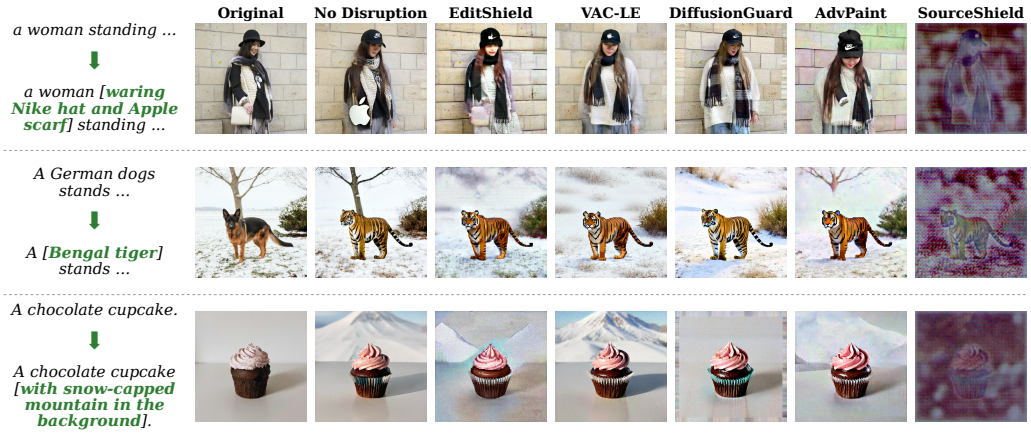


Figure 5: Visualizations of editing results on unseen source–target prompt pairs.

5.2 Effectiveness on Unseen Source-Target Prompts

Table 1 reports disruption performance under unseen source-target prompts. SourceShield achieves the best average results across 9 out of 10 metric cases, and the superiority is consistent across three unseen prompts. In addition, the superiority is not tied to a specific backbone architecture since SD3 is DiT-based while DS is U-Net-based. Visualizations of editing results in Fig. 5 confirm the dramatically stronger disruptions of our SourceShield.

Table 2: Disruption performance on unseen editing models. A→B denotes that perturbations are optimized against model A and then transferred to model B.

Methods	Metric									
	PSNR↓	LPIPS↑	FID↑	IR↓	CLIP-T↓	PSNR↓	LPIPS↑	FID↑	IR↓	CLIP-T↓
	DS → SD3					*SD3 → DS*				
EditShield	17.75	0.4071	83.8	1.19	33.98	24.01	0.3955	82.98	0.03	32.43
VAC-LE	17.91	0.4305	81.3	1.18	34.02	23.27	**0.4134**	83.49	-0.02	32.31
DiffusionGuard	17.92	0.3847	81.6	1.21	34.32	23.90	0.3878	83.68	0.12	32.18
AdvPaint	17.53	0.4209	84.9	1.29	34.26	23.31	0.3891	80.66	-0.05	31.52
SourceShield (Ours)	**17.27**	**0.4517**	**86.7**	**1.16**	**33.92**	**23.08**	0.4100	**86.77**	**-0.18**	**31.41**
	DS → FLUX					*SD3 → FLUX*				
EditShield	15.87	0.5306	109.1	1.31	35.04	18.87	0.3276	73.3	1.15	33.61
VAC-LE	16.12	0.5492	109.3	1.30	34.95	20.27	0.3530	72.9	1.33	**33.46**
DiffusionGuard	16.68	0.4967	97.1	1.32	34.92	18.52	0.3564	73.1	1.23	33.57
AdvPaint	16.05	0.5508	107.7	1.29	**34.81**	18.67	0.3787	71.9	**1.11**	33.91
SourceShield (Ours)	**15.61**	**0.5883**	**111.7**	**1.25**	34.89	**18.33**	**0.3827**	**75.7**	1.12	33.53

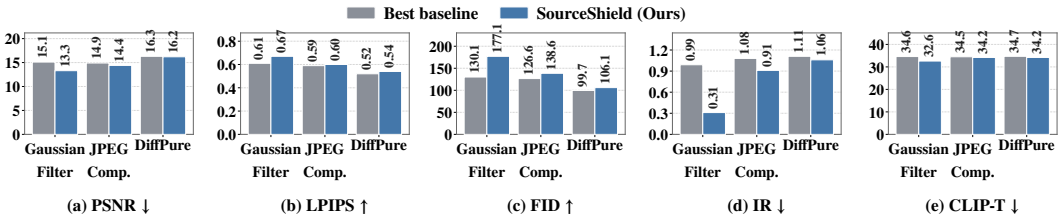


Figure 6: Resilience of different methods to three countermeasures: Gaussian filtering, JPEG compression, and DiffPure purification.

5.3 Transferability to Unseen Editing Models

Since a malicious editor may employ unseen models, we evaluate cross-model transferability in Table 2. SourceShield achieves the best results in 16 out of 20 metrics, indicating that source-conditioned accumulation disruption transfers across editing models. In particular, transfer to FLUX show that the surrogate model influences disruption behavior: DS→FLUX yields stronger output consistency disruption, while SD3→FLUX gives slightly stronger prompt faithfulness disruption.

5.4 Input-Level Imperceptibility

Table 3 evaluates the input-level imperceptibility of protected images. SourceShield achieves the lowest L_2-pixel/255 and the highest PSNR on both SD3 and DS, indicating strong pixel-level imperceptibility. It also obtains the best LPIPS on SD3 and the second-best LPIPS on DS, showing strong perceptual similarity between the input image and the protected image. Together with Table 1, these results show that SourceShield improves disruption without making the perturbation more visible, preserving input-level imperceptibility while shifting the trajectory of edited variables.

Table 3: Input-level imperceptibility of protected images.

Methods	Metric					
	L2-pixel/255 ↓		PSNR ↑		LPIPS ↓	
	SD3	DS	SD3	DS	SD3	DS
EditShield	13.34	12.40	25.64	26.27	0.5693	0.5514
VAC-LE	9.71	11.93	28.40	27.37	0.3863	0.4412
DiffusionGuard	9.67	11.60	28.11	27.49	0.3969	**0.3484**
AdvPaint	13.38	11.74	25.61	27.52	0.5157	0.4015
SourceShield	**9.48**	**11.28**	**28.60**	**27.89**	**0.3775**	0.3624

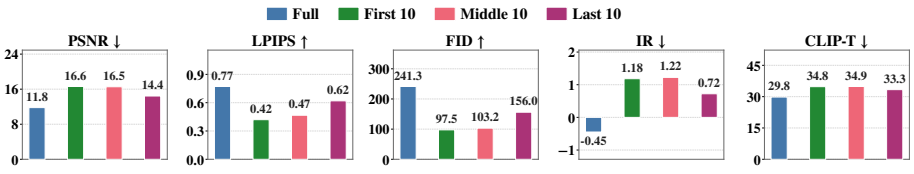


Figure 7: Ablation study on the selected solver timestep set $\mathcal{I}$.

5.5 Resilience to Countermeasures

We evaluate resilience to Gaussian filtering, JPEG compression, and DiffPure purification (Nie et al., 2022). Fig. 6 shows that SourceShield outperforms the strongest baseline in 27 out of 30 comparisons, indicating effectiveness after preprocessing and purification; full results are in Appendix E.1.

5.6 Ablation Studies

Chunk Size K. We study the chunk size K in two-stage chunk-wise gradient optimization. Table 4 reports peak GPU memory and optimization time under different chunk sizes. Processing all selected solver timesteps in one pass gives the lowest time but highest memory. Reducing K lowers peak memory from 67.23 GB to 27.28 GB at $K = 11$, with only a moderate time increase from 8.4 to 10.1 min/image. Further reducing K offers limited memory savings but increases recomputation overhead; we therefore use $K = 11$ by default.

Table 4: Ablation study on chunk size K.

K	Memory (GB) ↓	Time (min/image) ↓
9	22.49	10.3
11	27.28	10.1
16	34.1	9.8
Full	67.23	8.4

Objective Design. We compare SourceShield with two variants with other optimizations objectives. *Min-Accum.* minimizes the source-conditioned accumulation, testing the opposite optimization direction. *Step-Deviation* maximizes the average per-timestep deviation between source-conditioned velocities from the input image and the protected image. Table 5 shows that SourceShield performs best across all metrics, confirming the importance of directly maximizing the source-conditioned accumulation.

Table 5: Ablation study on different objectives.

Metrics	Min-Accun	Step-Deviation	SourceShield (Ours)
PSNR ↓	15.94	<u>14.40</u>	**11.79**
LPIPS ↑	0.5422	<u>0.6328</u>	**0.7715**
FID ↑	111.3	<u>151.7</u>	**241.3**
IR ↓	1.19	<u>0.99</u>	**-0.45**
CLIP-T ↓	34.70	<u>34.48</u>	**29.80**

Solver Timestep Set $\mathcal{I}$. We compare the full selected solver timestep set $\mathcal{I}$ with three 10-step windows: the first, middle, and last 10 solver timesteps. As shown in Fig. 7, the full set achieves the strongest disruption across all metrics, showing that SourceShield benefits from disrupting the complete source-conditioned accumulation rather than a local source-branch segment. Additional ablations on the perturbation budget ε are provided in Appendix F.1.

6 Conclusion and Outlook

In this work, we propose **SourceShield**, a novel disruption framework for inversion-free image editing that targets the editing trajectory in the source branch. By decomposing the trajectory of edited variables, SourceShield identifies the source-conditioned accumulation as a persistent and timestep-wise decomposable disruption point. Maximizing its magnitude shifts the velocity differences that drive edited variables, leading to strong degradation of unauthorized edited outputs while preserving input-level imperceptibility. We further introduce a two-stage chunk-wise gradient optimization to reduce activation memory. Extensive experiments show that SourceShield consistently outperforms existing baselines under unseen prompts, unseen editing models, and countermeasures. Looking forward, extending source-branch disruption to video editing is promising, where persistent disruption should also account for temporal consistency across frames.

AI USE STATEMENT

We used generative AI tools only for language polishing, grammar checking, translation assistance, LaTeX formatting, readability improvement, and consistency checks of terminology, equations, tables, figures, and references. We did not use generative AI tools to generate research ideas, design experiments, implement code, process data, produce proofs, conduct experiments, analyze results, or generate reported numbers. The authors reviewed all AI-assisted edits against the manuscript, source files, and experimental outputs, and take full responsibility for the accuracy, originality, and compliance of the final content.

REFERENCES

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *Arxiv*, 2022.

Black Forest Labs. Official weights of flux.1-dev. `https://huggingface.co/black-forest-labs/FLUX.1-dev`, 2024.

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

Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. In *ICCV*, 2023.

Ruoxi Chen, Haibo Jin, Yixin Liu, Jinyin Chen, Haohan Wang, and Lichao Sun. Editshield: Protecting unauthorized image editing by instruction-guided diffusion models. In *ECCV*, 2024.

June Suk Choi, Kyungmin Lee, Jongheon Jeong, Saining Xie, Jinwoo Shin, and Kimin Lee. Diffusionguard: A robust defense against malicious diffusion-based image editing. In *ICLR*, 2025.

Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *Arxiv*, 2022.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. In *Arxiv*, 2022.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

Seunghoo Hong, Geonho Son, Juhun Lee, and Simon S Woo. Dia: The adversarial exposure of deterministic inversion in diffusion models. In *ICCV*, 2025.

Bernd Jähne. Digital image processing. In *Springer*, 2005.

Joonsung Jeon, Woo Jae Kim, Suhyeon Ha, Sooel Son, and Sung-Eui Yoon. Advpaint: Protecting images from inpainting manipulation via adversarial attention disruption. In *Arxiv*, 2025.

Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. In *ICCV*, 2025.

Ang Li, Yichuan Mo, Mingjie Li, and Yisen Wang. Pid: Prompt-independent data protection against latent diffusion models. In *Arxiv*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *Arxiv*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *Arxiv*, 2022.

Ling Lo, Cheng Yu Yeo, Hong-Han Shuai, and Wen-Huang Cheng. Distraction is all you need: Memory-efficient image immunization against diffusion-based image editing. In *CVPR*, 2024.

Liangsi Lu, Xuhang Chen, Minzhe Guo, Shichu Li, Jingchao Wang, and Yang Shi. Chordedit: One-step low-energy transport for image editing. In *Arxiv*, 2026.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. In *Arxiv*, 2023.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Arxiv*, 2017.

Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion models for adversarial purification. In *Arxiv*, 2022.

OpenAI. ChatGPT (september 2024 version). `https://chat.openai.com/`, 2024.

Tarik Can Ozden, Ozgur Kara, Oguzhan Akcin, Kerem Zaman, Shashank Srivastava, Sandeep P Chinchali, and James M Rehg. Diffvax: Optimization-free image immunization against diffusion-based editing. In *ICLR*, 2026.

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 *ICML*, 2021.

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

Hadi Salman, Alaa Khaddaj, Guillaume Leclerc, Andrew Ilyas, and Aleksander Madry. Raising the cost of malicious ai-powered image editing. In *ICML*, 2023.

Shawn Shan, Jenna Cryan, Emily Wenger, Haitao Zheng, Rana Hanocka, and Ben Y Zhao. Glaze: Protecting artists from style mimicry by text-to-image models. In *USENIX Security*, 2023.

Geonho Son, Juhun Lee, and Simon S. Woo. Disrupting diffusion-based inpainters with semantic digression. In *IJCAI*, 2024.

Yiren Song, Pei Yang, Hai Ci, and Mike Zheng Shou. Idprotector: An adversarial noise encoder to protect against id-preserving image generation. In *CVPR*, 2025.

Stability AI. Official weights of stable diffusion 3 medium diffusers. `https://huggingface.co/stabilityai/stable-diffusion-3-medium-diffusers`, 2024.

Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *CVPR*, 2023.

Hanhui Wang, Yihua Zhang, Ruizheng Bai, Yue Zhao, Sijia Liu, and Zhengzhong Tu. Edit away and my face will not stay: Personal biometric defense against malicious generative editing. In *CVPR*, 2025.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. In *Arxiv*, 2025.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. In *NIPS*, 2023a.

Sihan Xu, Yidong Huang, Jiayi Pan, Ziqiao Ma, and Joyce Chai. Inversion-free image editing with natural language. In *Arxiv*, 2023b.

Haotian Xue, Chumeng Liang, Xiaoyu Wu, and Yongxin Chen. Toward effective protection against diffusion-based mimicry through score distillation. In *ICLR*, 2024.

Jiachen Yang, Yusong Wang, Yanmei Fang, Yunshu Dai, and Fangjun Huang. Variance as a catalyst: Efficient and transferable semantic erasure adversarial attack for customized diffusion models. In *ICML*, 2025.

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. In *Arxiv*, 2023.

Sung-Hoon Yoon, Minghan Li, Gaspard Beaudouin, Congcong Wen, Muhammad Rafay Azhar, and Mengyu Wang. Splitflow: Flow decomposition for inversion-free text-to-image editing. In *NIPS*, 2026.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Tianrui Zhu, Shiyi Zhang, Jiawei Shao, and Yansong Tang. Kv-edit: Training-free image editing for precise background preservation. In *ICCV*, 2025.

Appendix:

SourceShield: Source-Branch Adversarial Disruption for Inversion-Free Image Editing

APPENDIX CONTENTS

A Related Background **14**

A.1 Text-Guided Image Editing 14

A.2 Adversarial Disruption for Unauthorized Image Editing 14

B Preliminary **14**

B.1 Flow Matching and Rectified Flow Background 14

B.2 Inversion-Free Image Editing 15

C Accumulation Derivations **16**

C.1 Target-Conditioned Accumulation Derivation 16

C.2 Source-Conditioned Accumulation Decomposition 16

D Experimental Setups **16**

D.1 Detailed Dataset and Victim Models 16

D.2 Detailed Evaluation Metrics 17

D.3 Detailed Implementation Details 17

E Additional Results **18**

E.1 Detailed Countermeasure Results 18

E.2 Additional Qualitative Results 19

F Ablation Studies **19**

F.1 Perturbation Budget 19

G Limitations **19**

A Related Background

A.1 Text-Guided Image Editing

Text-guided image editing aims to modify an input image according to a target prompt while preserving prompt-irrelevant visual content. Existing methods can be broadly grouped into *training-based image editing*, *inversion-based image editing*, and *inversion-free image editing*. Training-based image editing learns editability by adapting model parameters or training additional conditioning modules. It includes full-parameter fine-tuning methods, such as InstructPix2Pix (Brooks et al., 2023) and Qwen-Image (Wu et al., 2025), personalization methods, such as DreamBooth (Ruiz et al., 2023) and Textual Inversion (Gal et al., 2022), and adapter-based methods, such as IP-Adapter (Ye et al., 2023) and ControlNet (Zhang et al., 2023). Despite their effectiveness, these methods are often computationally intensive, limiting their scalability in practical scenarios. Inversion-based image editing keeps model parameters fixed but relies on explicit inversion before editing. It typically follows a two-stage inversion-denoising process (Tumanyan et al., 2023; Hertz et al., 2022; Cao et al., 2023), which can accumulate inversion errors and introduce additional inference overhead. Inversion-free image editing bypasses both parameter adaptation and explicit inversion and directly updates edited variables through velocity differences (Xu et al., 2023b; Kulikov et al., 2025; Zhu et al., 2025; Yoon et al., 2026; Lu et al., 2026), improving efficiency and stability. InfEdit (Xu et al., 2023b) builds a inversion-free image editing framework on consistency models, while FlowEdit (Kulikov et al., 2025) extends this paradigm to Rectified Flow models by manipulating source- and target-conditioned velocity differences. Recent methods further strengthen inversion-free image editing: KV-Edit (Zhu et al., 2025) introduces KV caching for a DiT-based backbone, SplitFlow (Yoon et al., 2026) decomposes and aggregates flow components, and ChordEdit (Lu et al., 2026) explores one-step low-energy transport.

A.2 Adversarial Disruption for Unauthorized Image Editing

Existing disruption methods are mainly designed for training-based and inversion-based image editing, and broadly fall into early-stage disruption and output-stage disruption. Early-stage disruption perturbs intermediate signals before the final edited output is formed (Song et al., 2025; Chen et al., 2024; Wang et al., 2025; Shan et al., 2023; Yang et al., 2025; Li et al., 2024; Choi et al., 2025; Son et al., 2024; Jeon et al., 2025; Lo et al., 2024), assuming that early disturbances propagate through later solver timesteps. However, this stage-local assumption does not readily generalize to inversion-free image editing, where later solver updates may attenuate early disturbances and weaken their persistence along the trajectory of edited variables. Output-stage disruption directly optimizes the final edited output (Xue et al., 2024; Salman et al., 2023; Ozden et al., 2026; Hong et al., 2025), but only captures the final disruption effect and does not explicitly model how velocity differences accumulate across solver timesteps. As a result, existing disruption methods overlook the role of the editing trajectory in inversion-free image editing. In this work, we propose SourceShield, the first disruption method against inversion-free image editing. By disrupting the source-conditioned accumulation in the source branch, SourceShield persistently perturbs velocity differences and shifts the trajectory of edited variables to degrade edited outputs.

B Preliminary

B.1 Flow Matching and Rectified Flow Background

Flow Matching (FM) (Lipman et al., 2022; Albergo & Vanden-Eijnden, 2022; Liu et al., 2022) learns a continuous-time vector field that transports samples between two endpoint distributions. Let $x_t \in \mathbb{R}^d$ denote a trajectory variable at time $t \in [0, 1]$, where $t = 0$ corresponds to the data side and $t = 1$ corresponds to the noise side. Given endpoint distributions p_0 and p_1, FM defines a time-dependent path distribution p_t and learns $v_\theta : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ whose probability-flow ODE is

$$\frac{dx_t}{dt} = v_\theta(x_t, t). \tag{16}$$

Solving this ODE transports samples along the learned path; in generation, the learned field is typically integrated from a noisy variable toward the data side. The vector field therefore describes the

instantaneous direction in which the trajectory variable should move at each time. It is trained by matching a target velocity field $v^\star$:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\, x_t\sim p_t}\left[\|v_\theta(x_t,t) - v^\star(x_t,t)\|_2^2\right]. \tag{17}$$

Once the initial condition and solver are fixed, sampling is performed by deterministic ODE integration, which makes the learned trajectory directly analyzable.

Rectified Flow (RF) (Liu et al., 2022) is a widely used FM formulation that encourages straight trajectories between paired endpoints (x_0, x_1), with x_0 sampled from the data distribution and x_1 sampled from the noise distribution. It uses the linear interpolation

$$x_t = (1-t)x_0 + tx_1, \qquad v^\star(x_t,t) = x_1 - x_0. \tag{18}$$

For a given endpoint pair, the target velocity is the constant displacement $x_1 - x_0$. Accordingly, the training objective becomes

$$\mathcal{L}_{\mathrm{RF}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\,(x_0,x_1)}\left[\|v_\theta\left((1-t)x_0 + tx_1, t\right) - (x_1 - x_0)\|_2^2\right]. \tag{19}$$

For text-conditioned generation and editing, the vector field additionally takes a text condition φ and is written as $v_\theta(x_t, t, \varphi)$.

B.2 Inversion-Free Image Editing

Inversion-free image editing (Kulikov et al., 2025; Xu et al., 2023b) takes three inputs: an input image x^{in}, a source prompt φ^{src} describing the input image, and a target prompt φ^{tgt} specifying the desired edit. It produces an edited output $x_{t_0}^{\mathrm{edit}} = \mathcal{E}_\theta(x^{\mathrm{in}}, \varphi^{\mathrm{src}}, \varphi^{\mathrm{tgt}})$ aligned with the target prompt while preserving the content of the input image. It directly constructs source variables, target variables, and edited variables, and queries a pretrained generative backbone along them. Under the RF formulation, the source variable at time t is available in closed form:

$$x_t^{\mathrm{src}} = (1-t)x^{\mathrm{in}} + t\xi_t, \qquad \xi_t \sim \mathcal{N}(0, I). \tag{20}$$

Thus, the source variable x_t^{src} is obtained directly from the input image and a noise sample, which is the key property used by inversion-free image editing.

Let x_t^{tgt} denote the target variable at time t. The edited variable x_t^{edit} is related to the source and target variables by

$$x_t^{\mathrm{edit}} = x^{\mathrm{in}} + x_t^{\mathrm{tgt}} - x_t^{\mathrm{src}}. \tag{21}$$

This identity can be read as replacing the source-side displacement with the target-side displacement. Since x_t^{tgt} is not directly available at inference time, the editor constructs it from the current edited variable as

$$x_t^{\mathrm{tgt}} = x_t^{\mathrm{edit}} + x_t^{\mathrm{src}} - x^{\mathrm{in}}. \tag{22}$$

The pretrained generative backbone is then queried at the constructed target variable under the target prompt and at the source variable under the source prompt. The difference between these two velocity predictions gives the edit direction. The evolution of the edited variable is driven by the difference between the target-conditioned and source-conditioned velocities:

$$\frac{dx_t^{\mathrm{edit}}}{dt} = v_\theta^{\Delta}\left(x_t^{\mathrm{src}}, x_t^{\mathrm{tgt}}\right), \tag{23}$$

where

$$v_\theta^{\Delta}\left(x_t^{\mathrm{src}}, x_t^{\mathrm{tgt}}\right) = v_\theta^{\mathrm{tgt}}\left(x_t^{\mathrm{tgt}}, t, \varphi^{\mathrm{tgt}}\right) - v_\theta^{\mathrm{src}}\left(x_t^{\mathrm{src}}, t, \varphi^{\mathrm{src}}\right). \tag{24}$$

We call $v_\theta^{\mathrm{tgt}}(x_t^{\mathrm{tgt}}, t, \varphi^{\mathrm{tgt}})$ the *target-conditioned velocity* and $v_\theta^{\mathrm{src}}(x_t^{\mathrm{src}}, t, \varphi^{\mathrm{src}})$ the *source-conditioned velocity*. They form the target branch and the source branch, respectively. The edited variables are initialized at the largest solver timestep by $x_{t_T}^{\mathrm{edit}} = x^{\mathrm{in}}$. With an Euler solver, consecutive solver timesteps are updated by

$$x_{t_{i-1}}^{\mathrm{edit}} = x_{t_i}^{\mathrm{edit}} + \Delta t_i\, v_\theta^{\Delta}\left(x_{t_i}^{\mathrm{src}}, x_{t_i}^{\mathrm{tgt}}\right), \qquad \Delta t_i = t_{i-1} - t_i. \tag{25}$$

Let $\{t_i\}_{i=0}^{T}$ denote the ordered solver timesteps and define $\mathcal{T} = \{1, \ldots, T\}$. Eq. (25) is applied for $i \in \mathcal{T}$ until $t_0 = 0$, where the edited output $x_{t_0}^{\mathrm{edit}}$ is obtained.

C Accumulation Derivations

C.1 Target-Conditioned Accumulation Derivation

We derive why the target-conditioned accumulation is not a suitable target-prompt-independent and timestep-wise decomposable disruption point. For a protected image x^{adv}, the target-conditioned accumulation is

$$A^{\mathrm{tgt}}(x^{\mathrm{adv}};\varphi^{\mathrm{src}},\varphi^{\mathrm{tgt}}) = \sum_{i\in\mathcal{T}} \Delta t_i \, v_\theta^{\mathrm{tgt}}\left(x_{t_i}^{\mathrm{tgt}}(x^{\mathrm{adv}}), t_i, \varphi^{\mathrm{tgt}}\right). \tag{26}$$

It explicitly depends on the target prompt φ^{tgt}, which is unknown when constructing a protected image for unseen editing requests. Moreover, differentiating Eq. (26) gives

$$J_{x^{\mathrm{adv}}} A^{\mathrm{tgt}} = \sum_{i\in\mathcal{T}} \Delta t_i \frac{\partial v_\theta^{\mathrm{tgt}}\left(x_{t_i}^{\mathrm{tgt}}(x^{\mathrm{adv}}), t_i, \varphi^{\mathrm{tgt}}\right)}{\partial x_{t_i}^{\mathrm{tgt}}} J_{x^{\mathrm{adv}}} x_{t_i}^{\mathrm{tgt}}. \tag{27}$$

The target variable is constructed from the current edited variable:

$$x_{t_i}^{\mathrm{tgt}}(x^{\mathrm{adv}}) = x_{t_i}^{\mathrm{edit}}(x^{\mathrm{adv}}) + x_{t_i}^{\mathrm{src}}(x^{\mathrm{adv}}) - x^{\mathrm{adv}}, \tag{28}$$

and hence

$$J_{x^{\mathrm{adv}}} x_{t_i}^{\mathrm{tgt}} = J_{x^{\mathrm{adv}}} x_{t_i}^{\mathrm{edit}} + J_{x^{\mathrm{adv}}} x_{t_i}^{\mathrm{src}} - I. \tag{29}$$

Here $J_{x^{\mathrm{adv}}} x_{t_i}^{\mathrm{edit}}$ is produced recursively by earlier solver updates, e.g.,

$$J_{x^{\mathrm{adv}}} x_{t_{i-1}}^{\mathrm{edit}} = J_{x^{\mathrm{adv}}} x_{t_i}^{\mathrm{edit}} + \Delta t_i \left[J_{x^{\mathrm{adv}}} v_\theta^{\mathrm{tgt}}\left(x_{t_i}^{\mathrm{tgt}}, t_i, \varphi^{\mathrm{tgt}}\right) - J_{x^{\mathrm{adv}}} v_\theta^{\mathrm{src}}\left(x_{t_i}^{\mathrm{src}}, t_i, \varphi^{\mathrm{src}}\right)\right]. \tag{30}$$

Thus, differentiating the target-conditioned accumulation requires propagating gradients through the time-ordered trajectory of edited variables. This temporal coupling prevents A^{tgt} from being decomposed into independent per-timestep gradient contributions, leading to higher gradient-computation cost.

C.2 Source-Conditioned Accumulation Decomposition

We derive why the source-conditioned accumulation is target-prompt-independent and timestep-wise decomposable. For a protected image x^{adv}, the source variable at timestep t_i is constructed as

$$x_{t_i}^{\mathrm{src}}(x^{\mathrm{adv}}) = (1-t_i)x^{\mathrm{adv}} + t_i \xi_{t_i}, \qquad \xi_{t_i} \sim \mathcal{N}(0, I). \tag{31}$$

The source-conditioned accumulation is

$$A^{\mathrm{src}}(x^{\mathrm{adv}};\varphi^{\mathrm{src}}) = \sum_{i\in\mathcal{T}} \Delta t_i \, v_\theta^{\mathrm{src}}\left(x_{t_i}^{\mathrm{src}}(x^{\mathrm{adv}}), t_i, \varphi^{\mathrm{src}}\right). \tag{32}$$

Unlike A^{tgt}, this accumulation does not depend on the future target prompt φ^{tgt} or on the evolving edited variables. Each source variable is constructed independently from x^{adv} and its per-timestep noise. Therefore, differentiating Eq. (32) gives an additive Jacobian:

$$J_{x^{\mathrm{adv}}} A^{\mathrm{src}}(x^{\mathrm{adv}};\varphi^{\mathrm{src}}) = \sum_{i\in\mathcal{T}} \Delta t_i \, J_{x^{\mathrm{adv}}} v_\theta^{\mathrm{src}}\left(x_{t_i}^{\mathrm{src}}, t_i, \varphi^{\mathrm{src}}\right). \tag{33}$$

This decomposition enables independent per-timestep gradient computation and avoids backpropagation through the time-ordered trajectory of edited variables. Since A^{src} is subtracted from A^{tgt} in the trajectory decomposition, disrupting it persistently perturbs the velocity differences that drive the trajectory of edited variables.

D Experimental Setups

D.1 Detailed Dataset and Victim Models

We construct an inversion-free image editing dataset with 127 images adapted from existing image editing benchmarks used by FlowEdit (Kulikov et al., 2025) and DiffVAX (Ozden et al., 2026).

The dataset covers diverse categories, including humans, animals, and objects. For each image, we use ChatGPT (OpenAI, 2024) to generate four source-target prompt pairs, where the source prompt captions the input image and the target prompt specifies a distinct editing intent. Each image-prompt pair defines one editing instance. We use one prompt pair per image as the seen instance for protected image optimization and the remaining three pairs as unseen instances for evaluation, resulting in 127 seen and 381 unseen editing instances.

For victim models, we use Stable Diffusion 3 (SD3) (Stability AI, 2024) and DreamShaper (DS) (Luo et al., 2023) for white-box optimization. In the black-box transfer setting, protected images optimized on SD3 or DS are evaluated on different victim models, including SD3, DS, and FLUX.1-dev (FLUX) (Black Forest Labs, 2024).

D.2 Detailed Evaluation Metrics

We evaluate protected images from three perspectives: output consistency, prompt faithfulness, and input-level imperceptibility.

Output Consistency Metrics. To assess how the perturbation affects edited outputs, we compare the edited output generated from the input image with that generated from the protected image. We use PSNR (Jähne, 2005), LPIPS (Zhang et al., 2018), and FID (Heusel et al., 2017). PSNR measures pixel-level similarity between the two edited outputs, LPIPS evaluates perceptual similarity using deep visual features, and FID measures distribution-level discrepancy over the evaluation set. Stronger disruption corresponds to lower PSNR, higher LPIPS, and higher FID.

Prompt Faithfulness Metrics. We use ImageReward (IR) (Xu et al., 2023a) and CLIP text-image similarity (CLIP-T) (Radford et al., 2021) to measure whether the edited output from the protected image remains aligned with the target prompt. IR provides a human-aligned assessment of visual quality and semantic consistency, while CLIP-T computes text-image alignment in a shared embedding space. Lower IR and CLIP-T indicate stronger disruption of prompt faithfulness.

Input-Level Imperceptibility Metrics. To verify that the protected image remains visually close to the input image, we compare the input image and the protected image using L_2-pixel/255, PSNR (Jähne, 2005), and LPIPS (Zhang et al., 2018). Here, L_2-pixel/255 denotes the pixel-space ℓ_2 distance between the input and protected images after normalizing pixel values by 255. Lower L_2-pixel/255, higher PSNR, and lower LPIPS indicate less visible perturbation.

D.3 Detailed Implementation Details

Implementation Details of SourceShield. All experiments are conducted on an NVIDIA A100 GPU with 80 GB of memory, and all implementations are developed in PyTorch. For each victim model, we use its default editing and inference settings unless otherwise specified. We optimize protected images in pixel space using Projected Gradient Descent (PGD) (Madry et al., 2017) for 200 iterations with step size $\alpha = 1/255$ and perturbation budget $\varepsilon = 16/255$. The protected image is initialized with uniform noise within the perturbation budget and projected back to the ℓ_∞ ball after each update. We fix the random seed to 42 during optimization, and edit the clean image and protected image with the same inference seed for each evaluation instance. SourceShield uses only the source prompt from the seen editing instance and does not require any target prompt during optimization. For two-stage chunk-wise gradient optimization, we set the default chunk size to $K = 11$.

Implementation Details of Baselines. We compare with representative early-stage disruption baselines, including EditShield, VAC-LE, DiffusionGuard, and AdvPaint. For fair comparison, all baselines use the same perturbation budget $\varepsilon = 16/255$, 200 optimization iterations, and the same seen source-target prompt split as SourceShield. When a baseline requires a source prompt during optimization, we use the source prompt from the seen editing instance; when it requires a target prompt, we use the seen target prompt. Other method-specific hyperparameters follow the original papers or official implementations whenever available.

Table 6: Resilience of different methods to countermeasures, evaluated with unseen source-target prompt pair A.

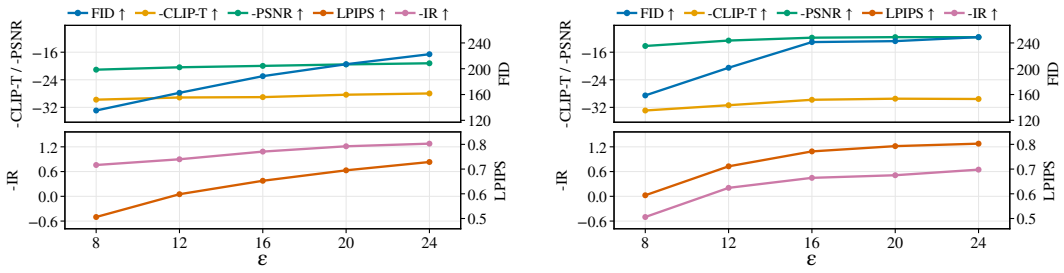
Countermeasure	Method	Metric				
		PSNR ↓	LPIPS ↑	FID ↑	IR ↓	CLIP-T ↓
Stable Diffusion 3 (SD3)						
Gaussian Filter	EditShield	15.13	0.6131	130.1	0.99	34.64
	VAC-LE	17.31	0.4115	89.1	1.27	34.86
	DiffusionGuard	16.29	0.4168	98.2	1.26	34.72
	AdvPaint	15.59	0.5871	118.3	1.05	35.22
	SourceShield (Ours)	**13.34**	**0.6652**	**177.1**	**0.31**	**32.56**
JPEG Comp.	EditShield	14.86	0.5967	126.6	1.08	34.49
	VAC-LE	16.57	0.5090	93.9	1.28	35.09
	DiffusionGuard	16.02	0.4780	101.1	1.27	34.83
	AdvPaint	15.33	0.5873	121.6	1.13	35.08
	SourceShield (Ours)	**14.41**	**0.6044**	**138.6**	**0.91**	**34.17**
DiffPure	EditShield	17.97	0.5182	96.55	1.19	34.82
	VAC-LE	17.11	0.4482	89.63	1.26	35.04
	DiffusionGuard	16.32	0.4301	98.8	1.23	34.71
	AdvPaint	16.43	0.5037	99.7	1.11	35.11
	SourceShield (Ours)	**16.29**	**0.5407**	**106.1**	**1.06**	**34.18**
DreamShaper (DS)						
Gaussian Filter	EditShield	20.97	**0.5859**	159.76	-0.60	30.64
	VAC-LE	22.65	0.3943	102.8	-0.32	31.12
	DiffusionGuard	23.07	0.4184	97.8	-0.28	31.01
	AdvPaint	21.29	0.4914	132.5	-0.71	30.02
	SourceShield (Ours)	**20.69**	<u>0.5819</u>	**163.4**	**-0.89**	**29.39**
JPEG Comp.	EditShield	20.69	**0.6214**	152.1	-0.58	30.52
	VAC-LE	22.17	0.4779	107.1	-0.32	31.23
	DiffusionGuard	22.81	0.4612	98.2	-0.27	30.86
	AdvPaint	21.12	0.5303	133.5	-0.67	30.32
	SourceShield (Ours)	**20.54**	<u>0.6156</u>	**167.2**	**-0.86**	**29.53**
DiffPure	EditShield	21.34	**0.6750**	136.9	-0.55	30.89
	VAC-LE	22.92	0.4050	97.2	-0.27	31.19
	DiffusionGuard	23.19	0.3856	84.8	-0.21	31.17
	AdvPaint	21.52	0.4698	118.9	-0.62	30.41
	SourceShield (Ours)	**21.05**	<u>0.5566</u>	**145.6**	**-0.74**	**30.02**

E Additional Results

E.1 Detailed Countermeasure Results

Table 6 provides the full countermeasure results on SD3 and DS. SourceShield consistently maintains strong disruption after Gaussian filtering, JPEG compression, and DiffPure purification. On SD3, SourceShield achieves the best results across all metrics under the three countermeasures, showing that its disruption remains effective even when the protected image is preprocessed before editing. On DS, SourceShield obtains the best PSNR, FID, IR, and CLIP-T under all countermeasures, and remains competitive on LPIPS. This indicates that the protected image preserves strong resilience across both DiT-based and U-Net-based backbones.

The results are especially meaningful under DiffPure, which is designed to remove adversarial perturbations through diffusion-based purification. SourceShield still achieves the highest FID on both SD3 and DS, and the lowest CLIP-T among all methods, demonstrating that the disrupted source-conditioned accumulation cannot be easily removed by image-level purification. These results further support the robustness of source-branch disruption under practical preprocessing scenarios.



(a) DS (b) SD3

Figure 8: Ablation study on the perturbation budget ε on DS and SD3.

E.2 Additional Qualitative Results

We provide more edited outputs under unseen target prompts on Stable Diffusion 3 and DreamShaper in Fig. 9 and Fig. 10, respectively.

F Ablation Studies

F.1 Perturbation Budget

We vary the perturbation budget ε to study the trade-off between input-level imperceptibility and disruption strength on DS and SD3. In Fig. 8, PSNR, CLIP-T, and IR are reported with negative signs so that larger values consistently indicate stronger disruption. Increasing ε allows SourceShield to introduce stronger perturbations to the source-conditioned accumulation, which more strongly shifts the velocity differences and degrades edited outputs. The disruption metrics improve consistently as the budget increases, but the gains become marginal at large ε, while input-level imperceptibility becomes harder to preserve. We therefore set $\varepsilon = 16/255$ by default, as it offers a favorable trade-off between imperceptibility and disruption strength across both victim models.

G Limitations

Although SourceShield shows strong disruption performance across unseen prompts, unseen editing models, and common countermeasures, it still has several limitations. First, SourceShield is developed for current inversion-free image editing pipelines that update edited variables through target- and source-conditioned velocity differences. If future editing paradigms use substantially different update rules or remove the source branch, the source-conditioned accumulation may become less effective as a disruption point. Second, SourceShield relies on a source prompt to construct the source-conditioned accumulation. Inaccurate or overly coarse source prompts may weaken the perturbation signal, suggesting that more robust source-prompt construction is an important direction for future work. Third, our countermeasure evaluation covers Gaussian filtering, JPEG compression, and DiffPure purification, but stronger adaptive purification or physical transformations may still reduce disruption effectiveness. Finally, SourceShield optimizes a protected image for each input image; despite the two-stage chunk-wise gradient optimization, further reducing per-image optimization cost would improve scalability.

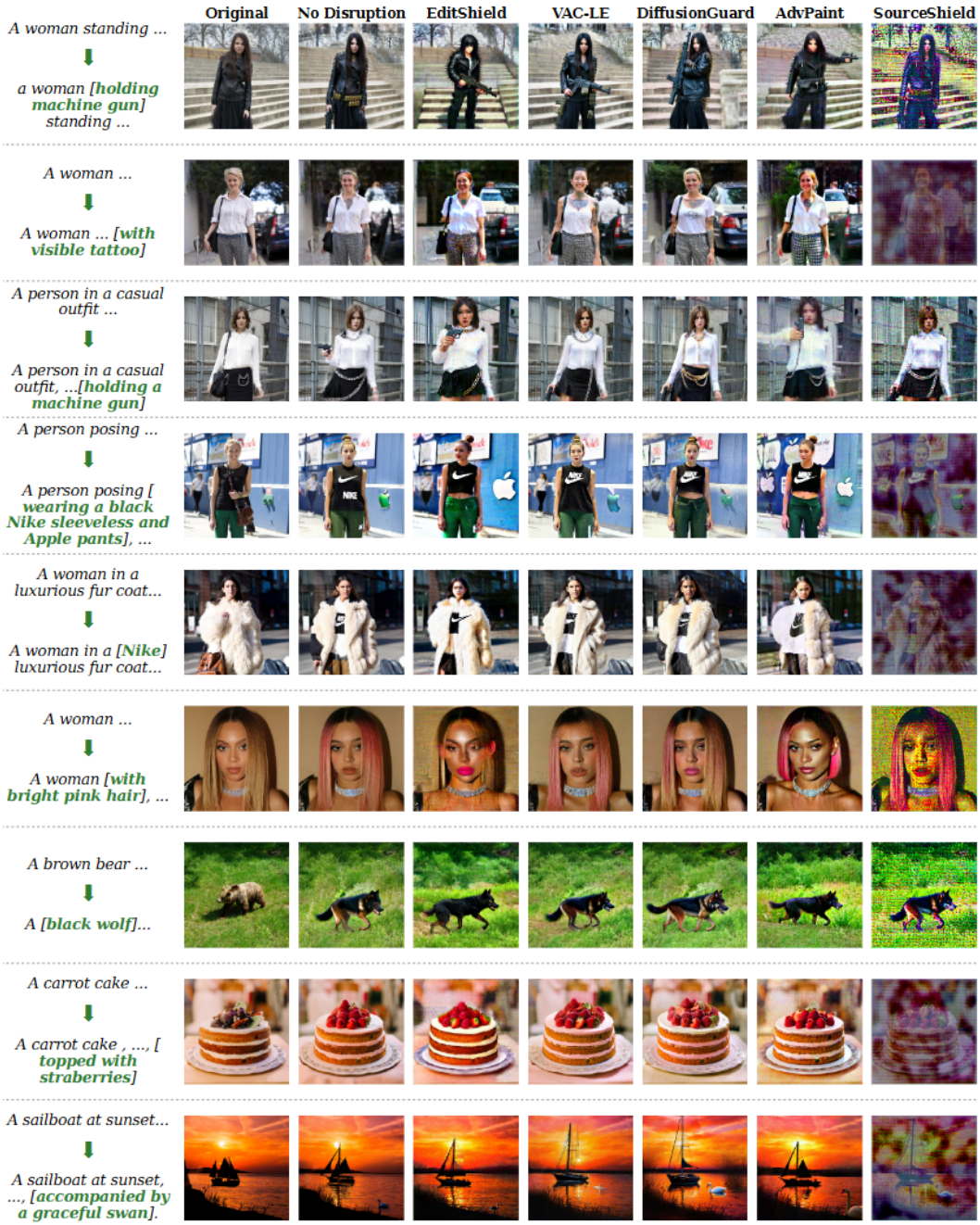


Figure 9: Additional qualitative comparisons under unseen target prompts on Stable Diffusion 3 (SD3).



Figure 10: Additional qualitative comparisons under unseen target prompts on DreamShaper (DS).