Precise Diffusion Inversion: Towards Novel Samples and Few-Step Models

Jing Zuo Luoping Cui Chuang Zhu Yonggang Qi ✉
School of Artificial Intelligence, Beijing University of Posts and Telecommunications
zuoj0723@gmail.com, {lpcui,czhu,qiyg}@bupt.edu.cn
✉ Corresponding author

Abstract

The diffusion inversion problem seeks to recover the latent generative trajectory of a diffusion model given a real image. Faithful inversion is critical for ensuring consistency in diffusion-based image editing. Prior works formulate this task as a fixed-point problem and solve it using numerical methods. However, achieving both accuracy and efficiency remains challenging, especially for few-step models and novel samples. In this paper, we propose *PreciseInv*, a general-purpose test-time optimization framework that enables fast and faithful inversion in as few as two inference steps. Unlike root-finding methods, we reformulate inversion as a learning problem and introduce a dynamic programming-inspired strategy to recursively estimate a parameterized sequence of noise embeddings. This design leverages the smoothness of the diffusion latent space for accurate gradient-based optimization and ensures memory efficiency via recursive subproblem construction. We further provide a theoretical analysis of *PreciseInv*'s convergence and derive a provable upper bound on its reconstruction error. Extensive experiments on COCO 2017, DarkFace, and a stylized cartoon dataset show that *PreciseInv* achieves state-of-the-art performance in both reconstruction quality and inference speed. Improvements are especially notable for few-step models and under distribution shifts. Moreover, precise inversion yields substantial gains in editing consistency for text-driven image manipulation tasks. Code is available at `https://github.com/panda7777777/PreciseInv`

1 Introduction

Large-scale pre-trained diffusion models [10, 17, 32, 33] have demonstrated strong generative capabilities in producing high-quality and diverse images. Built upon these models, many recent methods enable a wide range of image editing operations by guiding the generative process with user inputs, such as text prompts [4, 9, 16, 21, 22, 29], reference images [1, 6, 23, 25], and spatial masks [8, 28]. To ensure consistency and controllability in such edits, faithful inversion of real images into the diffusion process is essential [13].

Since the sampling process of diffusion models is inherently irreversible, mapping a real image back into the model domain remains a fundamental challenge. Existing methods on diffusion inversion can be divided into three major categories: forward optimization, backward optimization, and invertible samplers. Invertible samplers, such as EDICT [38], BDIA [43], and BELM [39], introduce explicit constraints into the sampling procedure to enable bidirectional generation. Backward optimization methods [21, 22, 29] fix the forward noising trajectory of a real image and optimize the reverse diffusion process to match this trajectory. Both types of methods improve reconstruction quality by modifying the denoising path, but often weaken the generative ability of the diffusion model. To address this issue, forward optimization methods estimate a noising trajectory that can be faithfully

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

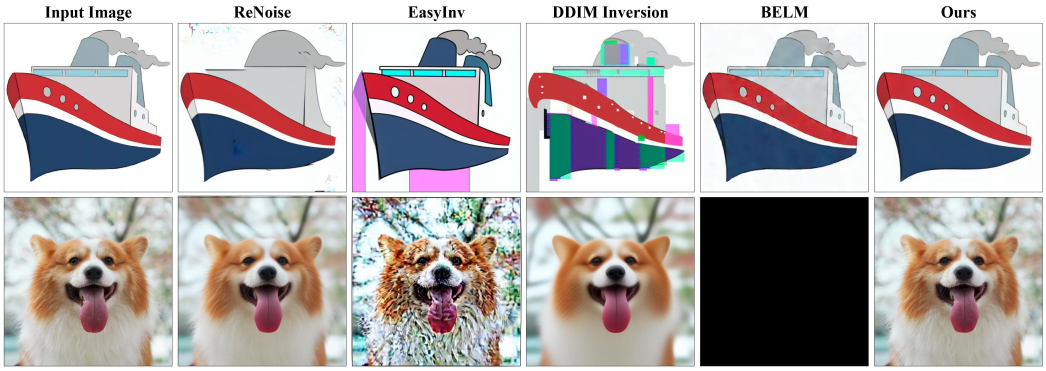

Figure 1: Qualitative comparison of diffusion inversion methods using SD v1.4 on challenging cases: novel sample (first row) and two-step inference (second row).

reversed by the diffusion model. For example, several methods [30, 34, 45] convert the problem into a fixed-point equation and solve it via Anderson acceleration [2], while ReNoise [13] refines an iterative renoising mechanism. However, these methods suffer from slow convergence and limited performance, especially under few-step inference or on novel samples, as illustrated in Fig. 1.

In this paper, we propose a general framework named *PreciseInv* that efficiently solves the diffusion inversion problem by progressively learning a parameterized noising trajectory. Specifically, we reformulate diffusion inversion as a learning problem and decompose it into T overlapping subproblems by leveraging the Markov property of the diffusion process. We then introduce a dynamic programming-inspired strategy [5] to recursively solve these subproblems. In each subproblem, *PreciseInv* learns a local noise embedding that approximates the optimal intermediate state along the inversion trajectory. Benefiting from the smooth transitions in the latent space [44], this gradient-based optimization converges more accurately than numerical solvers. Moreover, the recursive formulation is highly memory-efficient: inversion on the SD v1.4 base model requires only 3.43 GB of GPU memory, regardless of the number of inference steps. Extensive experiments show that *PreciseInv* achieves state-of-the-art reconstruction performance in both quality and efficiency (Section 4.1). The improvements are especially pronounced under few-step inference (e.g., $T{=}2, 4$) and on novel samples. Furthermore, precise inversion yields more semantically continuous latent representations (Fig. 5), which substantially improve consistency and controllability in text-driven editing tasks (Section 4.2).

Our main contributions are summarized as follows: (i) We propose *PreciseInv*, a general-purpose test-time optimization framework for diffusion inversion that enables fast and faithful image reconstruction under few-step inference and distribution shift. (ii) We reformalize diffusion inversion as a sequence of non-overlapping learning subproblems and introduce a dynamic programming strategy to solve them efficiently. (iii) We demonstrate that precise inversion yields smoother latent trajectories, which in turn improve consistency and controllability in text-driven image editing.

2 Background

2.1 Diffusion Models

Given samples from the data distribution $q(\mathbf{x}_0)$, diffusion models [17] define a generative process by learning to reverse a fixed Markovian forward process that progressively corrupts data by adding Gaussian noise. The forward process is formulated as:

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}), \quad q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\,\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I}), \tag{1}$$

where $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, $\{\mathbf{x}_t\}_{t=1}^T$ are latent variables, and $\{\alpha_t\}_{t=1}^T \in (0,1]^T$ is a predefined noise schedule. The marginal distribution at an arbitrary timestep t is given by:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}), \tag{2}$$

where $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$. The generative process is modeled as a parameterized Markov chain:

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)), \tag{3}$$

where $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$ and θ denotes the model parameters. In practice, a denoising neural network $\boldsymbol{\epsilon}_\theta$ is trained to predict $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$, and $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)$ is set to a constant σ_t according to α_t. The training objective minimizes a simplified variational bound:

$$\mathcal{L}_{\text{simple}}(\theta) := \mathbb{E}_{t \sim \mathcal{U}\{1,\ldots,T\},\, \mathbf{x}_0 \sim q(\mathbf{x}_0),\, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\mathbf{x}_t, t \right) \right\|^2 \right], \tag{4}$$

where

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\, \boldsymbol{\epsilon}. \tag{5}$$

Starting from $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$, the DDPM sampling procedure [17] iteratively applies the reverse step to generate $\mathbf{x}_0$ as follows:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\sqrt{1 - \alpha_t}}{\sqrt{1 - \alpha_{t-1}}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}, \tag{6}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. To reduce the number of inference steps, the DDIM sampler [36] converts Eq. (6) into a non-Markovian form:

$$\mathbf{x}_{t-1} = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}} \mathbf{x_t} + \left(\sqrt{\frac{1}{\alpha_{t-1}} - 1} - \sqrt{\frac{1}{\alpha_t} - 1} \right) \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t). \tag{7}$$

To further enhance the inference efficiency, recent efforts have focused on training few-step diffusion models, such as Consistency Models (CM) [37], Latent Consistency Models (LCM) [27], Adversarial Diffusion Distillation (ADD) models [35], Shortcut models [12] and MeanFlow [14].

2.2 Image Inversion in Diffusion Models

Inverting real images into the latent space of a pretrained diffusion model is critical for diffusion-based image editing methods. Given a real image $\mathbf{x}_0$, our goal is to estimate a corresponding noisy latent $\mathbf{x}_T$, such that $\mathbf{x}_0$ can be reconstructed by applying the reversed denoising trajectory to $\mathbf{x}_T$. Unfortunately, the neural network $\boldsymbol{\epsilon}_\theta$ do not inherently predict noise direction $\boldsymbol{\epsilon}$ from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$. DDIM Inversion [8] was among the earliest methods attempted to address this issue. Rewriting the Eq. (7), we obtain:

$$\mathbf{x}_t = \sqrt{\frac{\alpha_t}{\alpha_{t-1}}} \mathbf{x}_{t-1} - \sqrt{\frac{\alpha_t}{\alpha_{t-1}}} \left(\sqrt{\frac{1}{\alpha_{t-1}} - 1} - \sqrt{\frac{1}{\alpha_t} - 1} \right) \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t). \tag{8}$$

DDIM Inversion assumes $\boldsymbol{\epsilon}_\theta(\mathbf{x}_{t-1}, t-1) \approx \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ and iteratively applies Eq. (8) from $t = 1$ to $t = T$. However, the denoising trajectory is nonlinear, resulting in discrepancies between predicted $\hat{\mathbf{x}_t}$ and actual $\mathbf{x}_t$.

Subsequently, there are three main lines of work on diffusion inversion: modifying the reversed denoising trajectory, designing invertible samplers or optimizing the initial noise latent $\mathbf{x}_T$. A common practice of the former is to optimize the denoising process to better align with the trajectory obtained by DDIM inversion. For example, Null-Text Inversion [29] optimizes unconditional text embeddings to implicitly adjust the denoising trajectory, while Eta Inversion [22] uses a hyperparameter η to control the trade-off between the stochastic denoising trajectory and the vanilla DDIM Inversion trajectory. Moreover, a DDPM inversion method [21] was proposed, which rectifies the DDPM denoising process using a forward trajectory constructed by independently applying Eq.(5) at each timestep t. More recently, several invertible samplers [38, 39, 43] have been proposed for bidirectional generation. While, in general, manually modifying the denoising trajectory enhancing reconstruction quality, it inevitably disrupts the original generative process of the diffusion model.

To preserve the original generative process, another line of methods has been proposed to estimate a more accurate value of $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ in Eq. (8). For example, ReNoise [13] ensembles multiple DDIM Inversion results at each iteration. In addition, several methods reformulates the diffusion inversion

problem as a fixed-point problem and solves the problem by improving Anderson acceleration [2] methods. Let the estimated value of $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ be denoted as $\boldsymbol{\epsilon}_t^*$, then we have:

$$\boldsymbol{\epsilon}_t^* = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t). \tag{9}$$

Substituting Eq. (9) into Eq. (8), we obtain:

$$\mathbf{x}_t = \sqrt{\frac{\alpha_t}{\alpha_{t-1}}}\mathbf{x}_{t-1} - \sqrt{\frac{\alpha_t}{\alpha_{t-1}}}\left(\sqrt{\frac{1}{\alpha_{t-1}} - 1} - \sqrt{\frac{1}{\alpha_t} - 1}\right)\boldsymbol{\epsilon}_t^*. \tag{10}$$

By replacing $\mathbf{x}_t$ in Eq. (9) with Eq. (10), we deduce that:

$$\boldsymbol{\epsilon}_t^* = \boldsymbol{\epsilon}_\theta\left(\sqrt{\frac{\alpha_t}{\alpha_{t-1}}}\mathbf{x}_{t-1} - \sqrt{\frac{\alpha_t}{\alpha_{t-1}}}\left(\sqrt{\frac{1}{\alpha_{t-1}} - 1} - \sqrt{\frac{1}{\alpha_t} - 1}\right)\boldsymbol{\epsilon}_t^*, t\right). \tag{11}$$

To effectively solve Fixed-Point Eq.(11), AIDI[30] uniformly mixes the outputs from two adjacent iterations in vanilla Anderson acceleration, while EasyInv [45] increases the influence of $\mathbf{x}_0$ in estimating $\mathbf{x}_T^*$ by directly predicting $\mathbf{x}_t^*$ instead of $\boldsymbol{\epsilon}_t^*$. Despite recent progress, achieving precise and efficient inversion remains challenging, especially under few-step inference or when the inputs are novel, as illustrated in Fig. 1.

3 Method

3.1 Reformulating Diffusion Inversion as a Learning Problem

Given a pretrained diffusion model that defines a mapping $\mathcal{X} : \mathbf{x}_T \mapsto \mathbf{x}_{T-1} \mapsto \cdots \mapsto \mathbf{x}_0$ from latent space to image space, we hypothesize that the inverse mapping $\mathcal{X}^{-1} : \mathbf{x}_0 \mapsto \mathbf{x}_1 \mapsto \cdots \mapsto \mathbf{x}_T$ exhibits a smooth optimization landscape, owing to the smoothness of transitions in the latent space [44]. Based on this observation, we reformulate diffusion inversion as a learning problem. Let $\mathbf{x}_0$ be an observed image, and let $\boldsymbol{\epsilon}_T^*$ denote a learnable noise embedding. The objective is defined as:

$$\arg\min_{\boldsymbol{\epsilon}_T^*} \left\|\mathcal{X}(\mathbf{x}_T^*) - \mathbf{x}_0\right\|^2, \tag{12}$$

where

$$\mathbf{x}_T^* = \sqrt{\bar{\alpha}_T}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_T}\,\boldsymbol{\epsilon}_T^*. \tag{13}$$

3.2 Progressive Denoising Trajectory Learning

Directly optimizing (12) requires backpropagation through all T timesteps, which incurs prohibitive memory and computational costs. To address this, we exploit the Markov property of the diffusion process to decompose problem (12) into T subproblems and solve them efficiently using dynamic programming principles.

Subproblem Decomposition Let $\{\mathbf{x}_t\}_{t=1}^T$ denote latent variables along the denoising trajectory from $\mathbf{x}_0$ to $\mathbf{x}_T$, as indicated by the spherical markers along the blue arrow in Fig. 2. Since only $\mathbf{x}_0$ is observed, we adopt a bottom-up dynamic programming strategy to sequentially estimate latent states from $\mathbf{x}_1$ to $\mathbf{x}_T$. To this end, we define a collection of subproblems $\{\mathcal{P}(t)\}_{t=1}^T$, where each $\mathcal{P}(t)$ corresponds to the estimation of $\mathbf{x}_t$. Specifically,

$$\mathcal{P}(t) = \arg\min_{\boldsymbol{\epsilon}_t^*} \left\|\mathcal{X}(\mathbf{x}_t^*) - \mathbf{x}_0\right\|^2, \tag{14}$$

with

$$\mathbf{x}_t^* = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t^*, \tag{15}$$

where $\boldsymbol{\epsilon}_t^*$ is the learnable noise embedding at timestep $t \in \{1, 2, \ldots, T\}$.

Optimal Substructure The inverse mapping $\mathcal{X}^{-1} : \mathbf{x}_0 \mapsto \mathbf{x}_1 \mapsto \cdots \mapsto \mathbf{x}_T$ inherits the Markov structure of $\mathcal{X}$, implying that the optimal solution to each subproblem $\mathcal{P}(t)$ depends only on the solution to the previous subproblem $\mathcal{P}(t-1)$ for all $t \geq 2$. In particular, $\mathcal{P}(1)$ depends solely on the known observation $\mathbf{x}_0$. This recursive dependency establishes that the subproblems $\{\mathcal{P}(t)\}_{t=1}^T$ exhibit the optimal substructure property.

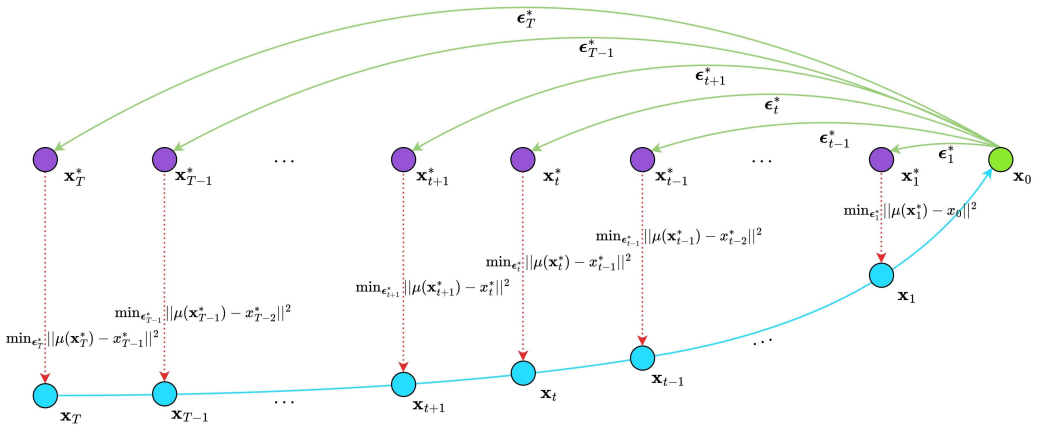

Figure 2: Overview of the proposed *PreciseInv*.

Overlapping Subproblems Each subproblem $\mathcal{P}(t)$ seeks a latent variable $\mathbf{x}_t$ such that the denoising trajectory from $\mathbf{x}_t$ accurately reconstructs $\mathbf{x}_0$. As adjacent subproblems $\mathcal{P}(t)$ and $\mathcal{P}(t-1)$ share this reconstruction objective and lie on successive steps of the trajectory, their optimization processes overlap. Specifically, the solution to $\mathcal{P}(t-1)$ influences and partially informs the optimization of $\mathcal{P}(t)$, resulting in substantial overlap across subproblems.

Bottom-Up Computation Without loss of generality, we adopt the DDIM sampling strategy. For convenience in the subsequent analysis, we reframe Eq. (7) as:

$$\mu(\mathbf{x}_t) = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}}\mathbf{x}_t + \left(\sqrt{\frac{1}{\alpha_{t-1}} - 1} - \sqrt{\frac{1}{\alpha_t} - 1}\right)\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t). \tag{16}$$

Substituting into Eq. (14), the first subproblem becomes:

$$\begin{aligned} \mathcal{P}(1) &= \arg\min_{\boldsymbol{\epsilon}_1^*} \left\|\mathcal{X}(\mathbf{x}_1^*) - \mathbf{x}_0\right\|^2 \\ &= \arg\min_{\boldsymbol{\epsilon}_1^*} \left\|\mu(\mathbf{x}_1^*) - \mathbf{x}_0\right\|^2. \end{aligned} \tag{17}$$

Exploiting the optimal substructure of the subproblems $\{\mathcal{P}(t)\}_{t=1}^T$, we derive a recursive formulation f that maps $\mathcal{P}(t-1)$ to $\mathcal{P}(t)$:

$$\begin{aligned} \mathcal{P}(t) &= \arg\min_{\boldsymbol{\epsilon}_t^*} \left\|\mathcal{X}(\mathbf{x}_t^*) - \mathbf{x}_0\right\|^2 \\ &= \arg\min_{\boldsymbol{\epsilon}_t^*} \left\|\mu(\mathbf{x}_t^*) - \mathbf{x}_{t-1}^*\right\|^2 \\ &= f(\mathcal{P}(t-1)), \end{aligned} \tag{18}$$

where $\mathbf{x}_{t-1}^*$ denotes the optimized solution obtained from $\mathcal{P}(t-1)$. Leveraging the recursive formulation f, we solve the subproblems $\{\mathcal{P}(t)\}_{t=1}^T$ iteratively from $t=1$ to $t=T$, as indicated by the red dotted arrows in Fig. 2. We define a convergence threshold η for $\mathcal{P}(t)$, and stop the optimization when $\|\mu(\mathbf{x}_t^*) - \mathbf{x}_{t-1}^*\|^2 < \eta$. In practice, we set η in the range of 10^{-2} to 10^{-5}. η and the inference step number T jointly control the trade-off between reconstruction accuracy and inference speed, as detailed in Table 5.

Optimization Procedure. The overall optimization algorithm of *PreciseInv* is presented in Algorithm 1. The procedure is memory-efficient since backpropagation is performed only over a single timestep. Moreover, *PreciseInv* is general and model-agnostic. We show its adaptability to different diffusion backbones (e.g., rectified flow [26]) and samplers (e.g., DDPM [17]) in Appendix A, and provide additional quantitative results in Appendix D.1.

Algorithm 1 *PreciseInv* for Diffusion Models with DDIM Sampler

Input: Real image $\mathbf{x}_0$, diffusion model $\boldsymbol{\epsilon}_\theta$, convergence threshold η, number of inference steps T
Output: Inverted latent $\mathbf{x}_T^*$
for $t = 1$ to T **do**
    Initialize $\boldsymbol{\epsilon}_t^* \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    if $t > 1$ **then**
        $\mathbf{x}_{t-1}^* \leftarrow \sqrt{\bar{\alpha}_{t-1}}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\boldsymbol{\epsilon}_{t-1}^*$
    else
        $\mathbf{x}_{t-1}^* \leftarrow \mathbf{x}_0$
    end if
    `// Apply a single DDIM Inversion step`
    $\boldsymbol{\epsilon}_t^* \leftarrow \frac{1}{\sqrt{1-\bar{\alpha}_t}}\left(\mu(\mathbf{x}_{t-1}^*) - \sqrt{\bar{\alpha}_t}\mathbf{x}_0\right)$ (μ defined in Eq. (16))
    $\mathbf{x}_t^* \leftarrow \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t^*$
    $\mathcal{L}_{\text{rec}} \leftarrow \left\|\mu(\mathbf{x}_t^*) - \mathbf{x}_{t-1}^*\right\|^2$
    while $\mathcal{L}_{\text{rec}} < \eta$ **do**
        $\mathbf{x}_t^* \leftarrow \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t^*$
        $\mathcal{L}_{\text{rec}} \leftarrow \left\|\mu(\mathbf{x}_t^*) - \mathbf{x}_{t-1}^*\right\|^2$
        $\boldsymbol{\epsilon}_t^* \leftarrow \boldsymbol{\epsilon}_t^* - \nabla\mathcal{L}_{\text{rec}}$
    end while
end for
return $\mathbf{x}_T^*$

3.3 Theoretical Analysis of Convergence

While the progressive optimization strategy above is conceptually appealing, its convergence has yet to be formally established. Here, we provide a theoretical proof to establish its convergence. We begin by stating two standard assumptions that are widely used in diffusion models [37].

Assumption 1. *The diffusion model $\epsilon_\theta(\mathbf{x}_t, t)$ is Lipschitz continuous in $\mathbf{x}_t$ with constant L_m, i.e., $\|\epsilon_\theta(\mathbf{x}_1, t) - \epsilon_\theta(\mathbf{x}_2, t)\| \le L_m \|\mathbf{x}_1 - \mathbf{x}_2\|$.*

Assumption 2. *Let $\mathcal{L}_t := \|\mu(\mathbf{x}_t) - \mathbf{x}_{t-1}\|^2$. The function $\mathcal{L}_t$ has L-Lipschitz continuous gradients with respect to $\mathbf{x}_t$, i.e., $\|\nabla\mathcal{L}_t(\mathbf{x}_t) - \nabla\mathcal{L}_t(\mathbf{x}_t')\| \le L\|\mathbf{x}_t - \mathbf{x}_t'\|$.*

Under these assumptions, standard non-convex optimization results [15] apply. With gradient descent using a fixed step size $\gamma < 2/L$ for each subproblem $\mathcal{P}(t)$, the local loss $\mathcal{L}_t$ decreases monotonically. After K iterations,

$$\min_{k\in[1,K]} \|\nabla\mathcal{L}_t^{(k)}\|^2 \le \mathcal{O}\left(\tfrac{1}{K}\right), \tag{19}$$

indicating convergence to a stationary point δ_t. Since each step minimizes a local reconstruction loss and the optimization proceeds recursively, the global loss $\mathcal{L}$ is non-increasing. Given that μ is Lipschitz in $\mathbf{x}_t$, the accumulated reconstruction error satisfies:

$$\|\mathcal{X}(\mathbf{x}_t) - \mathbf{x}_0\|^2 \le \sum_{t=1}^{T}\left(\prod_{j=1}^{t-1} L_m^{(j)}\right)\delta_t. \tag{20}$$

If $L_m^{(j)} \le L_m < 1$ (satisfies due to Assumption 1) and let $\delta = \max\{\delta_1, \cdots, \delta_T\}$, a geometric upper bound follows:

$$\|\mathcal{X}(\mathbf{x}_T) - \mathbf{x}_0\|^2 \le \delta\sum_{t=1}^{T} L_m^{t-1} \le \tfrac{\delta}{1-L_m}. \tag{21}$$

Theorem 1. *Under Assumptions 1 and 2, gradient descent on $\mathcal{P}(t)$ with step size $\gamma < 2/L$ converges monotonically to a stationary point of $\mathcal{L}_t$.*

Theorem 2. *Let $\Gamma(t) := \|\mathcal{X}(\mathbf{x}_t) - \mathbf{x}_0\|^2$. Under Assumption 1 and Theorem 1, $\Gamma(T)$ admits a geometric upper bound, i.e., $\Gamma(T) \le \delta/(1-L_m)$.*

Detailed proofs of Theorem 1 and 2 are provided in Appendix B.

3.4 Prompt-driven Image Editing

We apply *PreciseInv* to enable prompt-driven editing by replacing the inversion stage without modifying the editing pipeline. Let $\mathbf{x}_0$ denote the input image, and let c_s and c_t denote the source and target prompts, respectively. We perform inversion conditioned on c_s, following the progressive optimization strategy in Eq. (14). At each step t, we solve

$$\mathcal{P}_{\text{edit}}(t) = \arg\min_{\boldsymbol{\epsilon}_t^*} \left\| \mu_{c_s}(\mathbf{x}_t^*) - \mathbf{x}_{t-1}^* \right\|^2, \tag{22}$$

where $\mu_c(\cdot)$ generalizes Eq. (16) to conditional generation. This yields a latent $\mathbf{x}_T^*$ that faithfully reconstructs $\mathbf{x}_0$ under the conditioning of the source prompt.

To generate the edited image, we reuse $\mathbf{x}_T^*$ and apply DDIM sampling under classifier-free guidance (CFG) [18], where the target and source prompts act as positive and negative conditioning signals, respectively. The conditional mean of the guided denoising process is:

$$\mu_{c_{\text{pos}},\, c_{\text{neg}}}(\mathbf{x}_t) = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}}\, \mathbf{x}_t + \left(\sqrt{\frac{1}{\alpha_{t-1}} - 1} - \sqrt{\frac{1}{\alpha_t} - 1} \right) \Big[(1+\omega)\, \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, c_{\text{pos}}) - \omega\, \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, c_{\text{neg}}) \Big], \tag{23}$$

where $\mathcal{X}_{c_{\text{pos}},\, c_{\text{neg}}}$ denotes the deterministic denoising trajectory governed by Eq. (23). This pipeline procedure enables prompt-driven editing without task-specific designs and improves fidelity and localization by aligning the initial latent more precisely with the structure of the input image.

4 Experiments

In this section, we present experimental results with the aim of (i) demonstrating the state-of-the-art performance of *PreciseInv* for image reconstruction in terms of both quality and efficiency; (ii) verifying that faithful inversion enhances consistency in prompt-guided image editing; and (iii) providing insights into its local behaviors and interaction mechanisms.

4.1 Image Reconstruction

Datasets. We evaluate *PreciseInv* on three domains: LAION-aligned, low-light, and stylized. (i) The COCO 2017 [24] validation set contains 5,000 natural images with diverse everyday scenes. It serves as a close proxy to the LAION distribution used in training most text-to-image diffusion models. (ii) The DarkFace [41] validation set includes 6,089 nighttime images captured in real-world low-light conditions, exhibiting extreme visibility degradation. (iii) The Cartoon dataset consists of 722 stylized images collected from the internet. These samples exhibit abstract shapes, exaggerated structures, and vivid palettes, reflecting rare visual styles far from the training distribution.

Baselines. We compare our method with forward optimization methods, including *DDIM Inversion* [8], *ReNoise* [13], and *EasyInv* [45]; and invertible samplers, including *EDICT* [38], *BDIA* [43], and *BELM* [39]. Following prior work [45, 39], we don't compare with backward optimization methods, as they are not designed for recovering the original generative trajectory of diffusion models. We carefully tune hyperparameters for all methods.

Metrics. We evaluate reconstruction quality using LPIPS, SSIM, and RSNR, which respectively measure perceptual similarity, structural alignment, and signal fidelity. To assess efficiency, we also report the average inference time per image.

Results and Analysis. Table 1 reports quantitative results on COCO, DarkFace, and Cartoon datasets using the SD v1.4 base model. With a tight convergence threshold (η=10^{-5}), *PreciseInv* achieves the lowest LPIPS, highest SSIM, and best PSNR across all datasets. Despite the high quality, its inference time remains competitive. With a relaxed threshold (η=10^{-2}), *PreciseInv* still surpasses most methods in LPIPS, SSIM, and PSNR, while reducing inference time to 5.33 s. Among forward optimization methods, *EasyInv* and *ReNoise* underperform the 1000-step *DDIM Inversion*, which suggests that numerical or iterative methods suffer from a performance bottleneck. In contrast, our method breaks through this bottleneck by a large margin. In particular, on the stylized Cartoon dataset,

Table 1: Quantitative comparison of diffusion inversion methods for image reconstruction using SD v1.4 on COCO, DarkFace, and Cartoon datasets.

	Method	LPIPS (↓)			SSIM (↑)			PSNR (↑)			Time (s, ↓)
		COCO	Dark.	Cart.	COCO	Dark.	Cart.	COCO	Dark.	Cart.	Avg.
Invertible Samplers	EDICT	0.430	0.329	0.032	0.367	0.630	0.931	14.04	20.42	29.87	45.50
	BDIA	0.431	0.329	0.033	0.366	0.630	0.946	14.04	20.42	30.89	172.40
	BELM	0.431	0.331	0.041	0.366	0.617	0.947	14.03	20.43	29.97	9.00
Forward Optimizing	DDIM Inversion	0.118	<u>0.073</u>	0.145	0.714	<u>0.871</u>	0.879	24.67	30.89	23.50	41.30
	ReNoise	0.120	0.086	0.323	0.724	0.869	0.851	25.00	30.84	22.21	38.20
	EasyInv	0.210	0.210	0.165	0.666	0.780	0.883	23.13	27.60	21.95	23.60
	PreciseInv (η=10^{-2})	<u>0.104</u>	0.074	<u>0.025</u>	<u>0.737</u>	0.861	<u>0.952</u>	<u>25.13</u>	<u>31.00</u>	<u>29.71</u>	**5.33**
	PreciseInv (η=10^{-5})	**0.078**	**0.052**	**0.018**	**0.756**	**0.878**	**0.960**	**25.86**	**31.79**	**31.42**	<u>23.29</u>

Table 2: Quantitative comparison of diffusion inversion methods for image reconstruction using LCM-SD v1.5 and SDXL on the COCO dataset.

	LCM-SD v1.5				SDXL			
	LPIPS (↓)	SSIM (↑)	PSNR (↑)	Time (s, ↓)	LPIPS (↓)	SSIM (↑)	PSNR (↑)	Time (s, ↓)
BELM	0.449	0.362	13.99	<u>9.1</u>	–	–	–	–
DDIM Inversion	0.626	0.420	15.74	35.5	0.492	0.464	16.24	154.8
ReNoise	0.603	0.414	15.59	83.1	0.424	0.533	18.26	27.4
EasyInv	–	–	–	–	0.194	0.683	20.89	35.1
PreciseInv ($\eta = 10^{-2}$)	<u>0.103</u>	<u>0.753</u>	<u>25.55</u>	**8.83**	<u>0.185</u>	<u>0.763</u>	<u>27.00</u>	<u>7.43</u>
PreciseInv ($\eta = 10^{-5}$)	**0.083**	**0.775**	**26.45**	25.54	**0.080**	**0.854**	**30.93**	41.11

Table 3: Quantitative results of image reconstruction under the few-step inference setting for different inversion methods using SD v1.4 on the COCO dataset.

	$T = 2$			$T = 4$		
	LPIPS (↓)	SSIM (↑)	PSNR (↑)	LPIPS (↓)	SSIM (↑)	PSNR (↑)
BELM	0.832	0.112	8.19	0.431	0.366	14.03
DDIM Inversion	0.331	0.591	20.62	0.306	0.593	20.65
ReNoise	0.199	0.671	*22.92*	0.158	0.702	23.53
EasyInv	0.563	0.357	14.97	0.537	0.332	13.66
PreciseInv	**0.077**	**0.766**	**25.93**	**0.084**	**0.762**	**25.65**

the performance gains are more pronounced, yielding lower LPIPS (-0.127), higher SSIM ($+0.081$), and better PSNR ($+7.92$ dB). Moreover, different invertible samplers exhibit similar reconstruction quality. Compared to *EDICT* and *BDIA*, *BELM* significantly decreases inference time from 45.5 s to 9.00 s. Although they achieve remarkable success on the Cartoon dataset, their performance remains limited on COCO and DarkFace. By comparison, our method shows robustness across data types and further improves overall efficiency.

For generality, we also present the reconstruction results on the COCO dataset using the few-step model LCM-SD v1.5 and the high-resolution model SDXL in Table 2. *PreciseInv* consistently achieves the best reconstruction quality across the two base models. With η=10^{-2}, it already significantly outperforms all baselines while maintaining fast inference (8.83 s for LCM-SD v1.5 and 7.43 s for SDXL). When using a tighter threshold (η=10^{-5}), the performance of *PreciseInv* is further improved. These results demonstrate that our method is general across different model architectures.

As shown in Table 3, we report the reconstruction performance of different inversion methods under the challenging few-step setting ($T = 2, 4$). *ReNoise* exhibits stronger few-step inference capability compared to other baselines. Nevertheless, our method achieves substantially better reconstruction quality across all metrics.

4.2 Prompt-driven Image Editing

Here, we present the results of *PreciseInv* on prompt-driven image editing, as shown in Fig. 3, and compare its editing performance with existing methods in Fig. 4. We demonstrate that *PreciseInv* enables the preservation of fine-grained details that are irrelevant to the editing prompt across a wide range of editing scenarios. For example, when editing a down jacket into a leather jacket, the original fabric folds are well preserved (left example of the first row in Fig. 3). Furthermore, compared to

Figure 3: **PreciseInv** enables consistent and controllable image editing across diverse scenarios. From top left to bottom right: **(1) texture editing** (*down* → *leather*), **(2) color editing** (*light* → *dark*), **(3) object transition** (*fox* → *wolf*), and **(4) style transfer** (*photo* → *sketch*). In each case, our method preserves non-edited attributes such as identity, pose, or background, demonstrating high fidelity in semantic consistency.

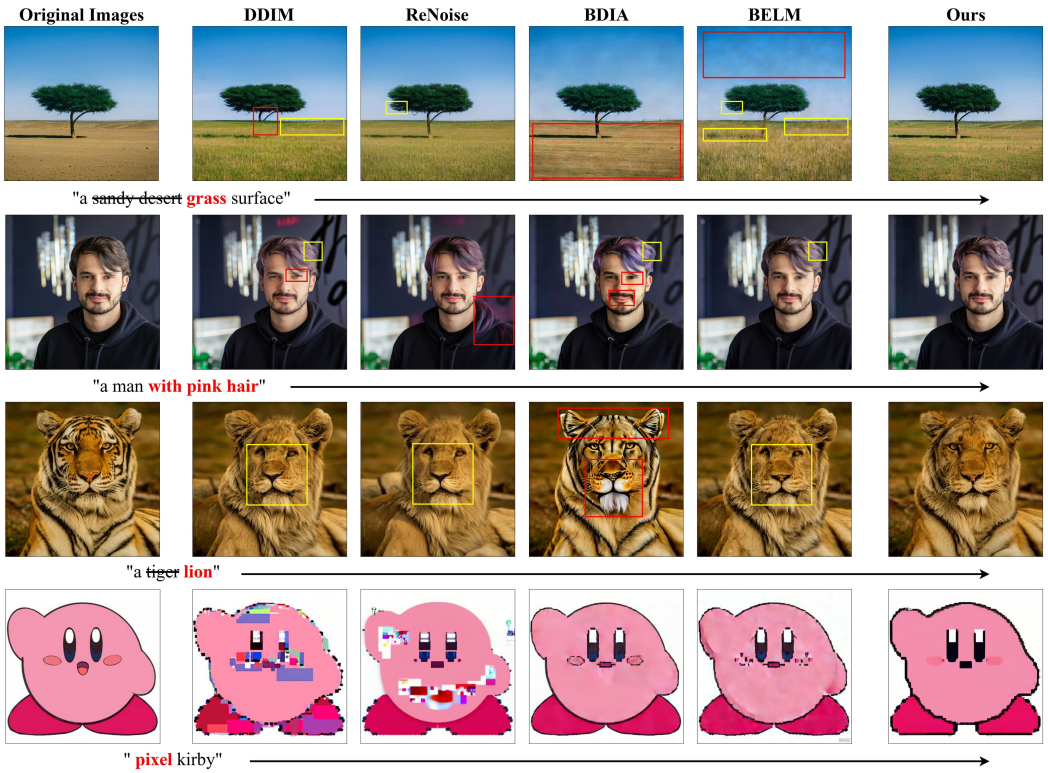

Figure 4: Comparison of different image editing results on prompt-driven image editing task.

existing methods, *PreciseInv* enables more faithful and controllable prompt-based editing. As shown in the first row of Fig. 4, our method better preserves both the tree structure and sky consistency during the texture transformation from desert to grassland, while baseline methods introduce sky artifacts (e.g., BELM) or distort tree details (e.g., ReNoise).

Notably, the purpose of these experiments is not to position *PreciseInv* as a state-of-the-art image editing method. As a generic inversion method without editing-specific design, it is inherently not comparable to advanced pipelines that involve domain-specific training [20, 40], attention manipulation [11, 16, 31, 42], or test-time optimization [7, 19]. Rather, *PreciseInv* serves as a foundation that can be integrated with such techniques. Incorporating precise inversion into editing-oriented frameworks is a promising direction for future work.

Figure 5: Interpolating between leftmost and rightmost images with spherical linear interpolation.

To further explore why *PreciseInv* enables controllable and faithful editing, we perform an interpolation-based diagnostic experiment, as shown in Fig. 5. Specifically, we first select two similar real images and apply different inversion methods to recover their corresponding latents. Then, we perform spherical linear interpolation (slerp) between the two inverted latents and reconstruct all intermediate states using the same diffusion model. As shown in Fig. 5, *PreciseInv* produces a smooth and semantically meaningful transition, maintaining structural coherence and identity consistency throughout the interpolation path. In contrast, DDIM and EasyInv often yield abrupt domain shifts or introduce foreign objects, indicating a drift away from the correct semantic manifold. BELM fails to preserve visual coherence due to accumulated noise artifacts, while ReNoise partially retains structure but suffers from geometric degradation. These findings suggest that accurate inversion leads to latents that better respect the geometry of the diffusion space, thereby enabling more localized, stable, and prompt-aligned edits.

5 Conclusion

We propose *PreciseInv*, a general-purpose test-time optimization framework that enables fast and faithful inversion for pre-trained diffusion models. Unlike prior fixed-point methods, we reformulate diffusion inversion as a progressive learning problem and introduce a dynamic programming-inspired strategy to solve it effectively. Theoretical analysis shows that *PreciseInv* converges under mild assumptions. Extensive experiments across diverse data domains and model architectures demonstrate that the proposed method achieves state-of-the-art performance in image reconstruction. Notably, the performance gains are especially pronounced under few-step settings and on novel samples. Moreover, we verify the effectiveness of *PreciseInv* in prompt-driven image editing tasks and further analyze its underlying mechanism. In future, we attempt to integrate *PreciseInv* with diverse editing techniques to explore a unified, controllable, and consistency-aware framework for image editing.

Acknowledgment

This work was supported by the Hainan Provincial Joint Project of Li'an International Education Innovation Pilot Zone, Grant No.624LALH008, NSFC under No.61601042, and the Program for Youth Innovative Research Team of BUPT under No. 2023QNTD02.

References

[1] Yuval Alaluf, Daniel Garibi, Or Patashnik, Hadar Averbuch-Elor, and Daniel Cohen-Or. Cross-image attention for zero-shot appearance transfer. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–12, 2024.

[2] Donald G Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12(4):547–560, 1965.

[3] Jhon Atchison and Sheng M Shen. Logistic-normal distributions: Some properties and uses. *Biometrika*, 67(2):261–272, 1980.

[4] 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, 2024.

[5] Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.

[6] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18392–18402, 2023.

[7] Jooyoung Choi, Yunjey Choi, Yunji Kim, Junho Kim, and Sungroh Yoon. Custom-edit: Text-guided image editing with customized diffusion models. *arXiv preprint arXiv:2305.15779*, 2023.

[8] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: diffusion-based semantic image editing with mask guidance. In *International Conference on Learning Representations*, 2023.

[9] Wenkai Dong, Song Xue, Xiaoyue Duan, and Shumin Han. Prompt tuning inversion for text-driven image editing using diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7430–7440, 2023.

[10] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[11] Kunyu Feng, Yue Ma, Bingyuan Wang, Chenyang Qi, Haozhe Chen, Qifeng Chen, and Zeyu Wang. Dit4Edit: Diffusion Transformer for image editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 2969–2977, 2025.

[12] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[13] Daniel Garibi, Or Patashnik, Andrey Voynov, Hadar Averbuch-Elor, and Daniel Cohen-Or. Renoise: real image inversion through iterative noising. In *European Conference on Computer Vision*, pages 395–413, 2024.

[14] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

[15] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.

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

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

[18] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

[19] Jiancheng Huang, Yifan Liu, Jin Qin, and Shifeng Chen. Kv inversion: Kv embeddings learning for text-conditioned real image action editing. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pages 172–184. Springer, 2023.

[20] Nisha Huang, Yuxin Zhang, Fan Tang, Chongyang Ma, Haibin Huang, Weiming Dong, and Changsheng Xu. Diffstyler: Controllable dual diffusion for text-driven image stylization. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[21] Inbar Huberman-Spiegelglas, Vladimir Kulikov, and Tomer Michaeli. An edit friendly ddpm noise space: inversion and manipulations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12469–12478, 2024.

[22] Wonjun Kang, Kevin Galim, and Hyung Il Koo. Eta inversion: designing an optimal eta function for diffusion-based real image editing. In *European Conference on Computer Vision*, pages 90–106. Springer, 2024.

[23] Kuan Heng Lin, Sicheng Mo, Ben Klingher, Fangzhou Mu, and Bolei Zhou. Ctrl-x: Controlling structure and appearance for text-to-image generation without guidance. *Advances in Neural Information Processing Systems*, 37:128911–128939, 2024.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pages 740–755. Springer, 2014.

[25] Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Meiling Wang, Xin Li, Zhengxing Sun, Qian Li, and Errui Ding. Adaattn: Revisit attention mechanism in arbitrary neural style transfer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6649–6658, 2021.

[26] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[27] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

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

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

[30] Zhihong Pan, Riccardo Gherardi, Xiufeng Xie, and Stephen Huang. Effective real image editing with accelerated iterative diffusion inversion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15912–15921, 2023.

[31] Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–11, 2023.

[32] 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.

[33] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[34] Dvir Samuel, Barak Meiri, Haggai Maron, Yoad Tewel, Nir Darshan, Shai Avidan, Gal Chechik, and Rami Ben-Ari. Lightning-fast image inversion and editing for text-to-image diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[35] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pages 87–103. Springer, 2024.

[36] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[37] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 32211–32252, 2023.

[38] Bram Wallace, Akash Gokul, and Nikhil Naik. Edict: Exact diffusion inversion via coupled transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22532–22541, 2023.

[39] Fangyikang Wang, Hubery Yin, Yue-Jiang Dong, Huminhao Zhu, Hanbin Zhao, Hui Qian, Chen Li, et al. Belm: Bidirectional explicit linear multi-step sampler for exact inversion in diffusion models. *Advances in Neural Information Processing Systems*, 37:46118–46159, 2024.

[40] Zhizhong Wang, Lei Zhao, and Wei Xing. Stylediffusion: Controllable disentangled style transfer via diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7677–7689, 2023.

[41] Chen Wei, Wenjing Wang, Wenhan Yang, and Jiaying Liu. Deep retinex decomposition for low-light enhancement. In *British Machine Vision Conference (BMVC)*, 2018.

[42] Shuai Yang, Yifan Zhou, Ziwei Liu, and Chen Change Loy. Fresco: Spatial-temporal correspondence for zero-shot video translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8703–8712, June 2024.

[43] Guoqiang Zhang, Jonathan P Lewis, and W Bastiaan Kleijn. Exact diffusion inversion via bidirectional integration approximation. In *European Conference on Computer Vision*, pages 19–36. Springer, 2024.

[44] Zijian Zhang, Zhou Zhao, and Zhijie Lin. Unsupervised representation learning from pre-trained diffusion probabilistic models. *Advances in neural information processing systems*, 35:22117–22130, 2022.

[45] Ziyue Zhang, Mingbao Lin, Shuicheng Yan, and Rongrong Ji. Easyinv: toward fast and better ddim inversion. *arXiv preprint arXiv:2408.05159*, 2024.

Appendix & Supplementary Materials

Contents

A Formulations

In this section, we extend the formulation of *PreciseInv* to different combinations of base models and samplers, highlighting the generality and adaptability of our approach.

A.1 Formulation of Stochastic Sampler: DDPM

We begin with the stochastic DDPM sampler [17]. For clarity, we first restate its sampling step:

$$\mu(\mathbf{x}_t, \mathbf{z}_t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{1-\alpha_{t-1}}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}_t. \tag{24}$$

The added noise term $\sigma_t \mathbf{z}_t$ in Eq. (24) complicates the inversion task, as deterministic reconstruction from $\mathbf{x}_t$ is no longer guaranteed. To overcome this, we treat $\mathbf{z}_t$ as an additional learnable variable. Specifically, we modify each subproblem $\mathcal{P}(t)$ (defined in Eq. (14)) to jointly optimize the noise embedding $\boldsymbol{\epsilon}_t^*$ and the stochastic component $\mathbf{z}_t^*$:

$$\mathcal{P}(t) = \arg\min_{\boldsymbol{\epsilon}_t^*, \mathbf{z}_t^*} \left\| \mathcal{X}(\mathbf{x}_t^*, \mathbf{z}_t^*) - \mathbf{x}_0 \right\|^2, \tag{25}$$

where

$$\mathbf{x}_t^* = \sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\, \boldsymbol{\epsilon}_t^*. \tag{26}$$

This formulation preserves the dynamic programming structure of our method. In particular, the optimal substructure property remains intact, with each subproblem $\mathcal{P}(t)$ dependent only on the solution to the previous subproblem $\mathcal{P}(t-1)$:

$$\mathcal{P}(t) = f(\mathcal{P}(t-1)) = \begin{cases} \arg\min_{\boldsymbol{\epsilon}_1^*, \mathbf{z}_1^*} \left\| \mu(\mathbf{x}_1^*, \mathbf{z}_1^*) - \mathbf{x}_0 \right\|^2, & t = 1, \\ \arg\min_{\boldsymbol{\epsilon}_t^*, \mathbf{z}_t^*} \left\| \mu(\mathbf{x}_t^*, \mathbf{z}_t^*) - \mathbf{x}_{t-1}^* \right\|^2, & t > 1. \end{cases} \tag{27}$$

By learning both the latent noise and the stochastic perturbations at each step, our method generalizes to stochastic samplers and enables accurate inversion under the DDPM setting.

A.2 Formulation of Rectified Flow Models: SD3

Preliminary. Rectified Flow [26] defines score-based diffusion modeling by learning a velocity field over a linear interpolation path between the data distribution $q(\mathbf{x}_0)$ and the Gaussian prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The forward process is formulated as:

$$\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\boldsymbol{\epsilon}, \quad t \in [0, 1]. \tag{28}$$

A neural network learns a time-dependent velocity field $\mathbf{v}_\theta(\mathbf{x}_t, t)$ by minimizing the following training objective:

$$\mathcal{L}_{\text{RF}}(\theta) := \mathbb{E}_{t \sim p(t),\, \mathbf{x}_0 \sim q(\mathbf{x}_0),\, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\left\| \boldsymbol{\epsilon} - \mathbf{v}_\theta(\mathbf{x}_t, t) \right\|^2 \right]. \tag{29}$$

The backward (generative) process recovers $\mathbf{x}_0$ from a sample $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ by solving the following ordinary differential equation (ODE) in reverse time:

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}_\theta(\mathbf{x}_t, t). \tag{30}$$

Unlike discrete-time diffusion models such as DDPM and DDIM, Rectified Flow does not define an explicit update rule. Instead, the reverse trajectory is obtained by numerically integrating Eq. (30), typically via Euler discretization:

$$\mathbf{x}_{t-\Delta t} \approx \mathbf{x}_t - \Delta t \cdot \mathbf{v}_\theta(\mathbf{x}_t, t). \tag{31}$$

SD3 [10] refines Rectified Flow to enable large-scale pretraining for text-to-image generation. First, instead of adopting a uniform timestep schedule, it samples timesteps from a logit-normal [3] distribution defined as:

$$\pi_{\ln}(t; m, s) = \frac{1}{s\sqrt{2\pi}} \frac{1}{t(1-t)} \exp\left(-\frac{(\text{logit}(t) - m)^2}{2s^2} \right). \tag{32}$$

This design allows SD3 to flexibly control the bias and width of the timestep distribution. It further introduces a multimodal diffusion transformer that unifies cross-attention and self-attention mechanisms. This results in improved alignment between text and image modalities.

***PreciseInv* for the SD3 Model** We further extend *PreciseInv* to the continuous-time rectified flow model, SD3. To construct a tractable denoising trajectory, we first sample a sequence of timesteps $\{t_1, t_2, \ldots, t_T\}$ from the logit-normal distribution $\pi_{\ln}(i; m, s)$ defined in Eq. (32). These sampled timesteps discretize the continuous-time trajectory into a sequence of latent variables $\{\mathbf{x}_{t_0}, \mathbf{x}_{t_1}, \ldots, \mathbf{x}_{t_T}\}$, where each $\mathbf{x}_{t_i}$ is recursively approximated by Euler integration:

$$\mathbf{x}_{t_{i-1}} = \mathbf{x}_{t_i} - (t_i - t_{i-1}) \cdot v_\theta(\mathbf{x}_{t_i}, t_i), \tag{33}$$

with $\mathbf{x}_{t_T} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $i \in \{T, T-1, \ldots, 2\}$.

Following Section 3, we define T subproblems $\{\mathcal{P}(t_i)\}_{i=1}^{T}$, each targeting a local reconstruction step along the denoising path $\mathbf{x}_{t_T} \mapsto \cdots \mapsto \mathbf{x}_{t_1}$. For each t_i, we parameterize the latent variable $\mathbf{x}^*_{t_i}$ with a learnable noise embedding $\boldsymbol{\epsilon}^*_{t_i}$ as:

$$\mathbf{x}^*_{t_i} = (1 - t_i)\, \mathbf{x}_0 + t_i\, \boldsymbol{\epsilon}^*_{t_i}. \tag{34}$$

Each subproblem $\mathcal{P}(t_i)$ seeks to minimize the reconstruction error between the estimated and true signal:

$$\mathcal{P}(t_i) = \arg\min_{\boldsymbol{\epsilon}^*_{t_i}} \left\| \mathcal{X}(\mathbf{x}^*_{t_i}) - \mathbf{x}_0 \right\|_2, \tag{35}$$

where $\mathcal{X}$ denotes the Euler-discretized backward trajectory from $\mathbf{x}_{t_T}$ to $\mathbf{x}_{t_1}$. For clarity, we rewrite the Euler step in Eq. (31) as:

$$\mu(\mathbf{x}_t, t_i, t_{i-1}) = \mathbf{x}_t - (t_i - t_{i-1}) \cdot \mathbf{v}_\theta(\mathbf{x}_t, t_i). \tag{36}$$

With this notation, each subproblem can be reformulated recursively as:

$$\mathcal{P}(t_i) = \begin{cases} \arg\min_{\boldsymbol{\epsilon}^*_{t_1}} \left\| \mu(\mathbf{x}^*_{t_1}, t_1, 0) - \mathbf{x}_0 \right\|^2, & i = 1, \\ \arg\min_{\boldsymbol{\epsilon}^*_{t_i}} \left\| \mu(\mathbf{x}^*_{t_i}, t_i, t_{i-1}) - \mathbf{x}^*_{t_{i-1}} \right\|^2, & i > 1. \end{cases} \tag{37}$$

By progressively solving these subproblems from $\mathcal{P}(t_1)$ to $\mathcal{P}(t_T)$, *PreciseInv* achieves a precise inversion of the image $\mathbf{x}_0$ into the SD3 latent space $\mathbf{x}_1$.

B Proofs

We provide formal proofs for the theoretical results presented in Section 3.

B.1 Proof of Theorem 1

Proof. For the function $\mathcal{L}_t(\mathbf{x}_t)$ with L-Lipschitz continuous gradients (Assumption 2), we define the gradient-based update as:

$$\mathbf{x}_t^{(k+1)} = \mathbf{x}_t^{(k)} - \gamma \nabla_{\mathbf{x}_t} \mathcal{L}_t(\mathbf{x}_t^{(k)}), \tag{38}$$

where k denotes the iteration index and γ is the learning rate. By the smoothness lemma [15], for any $\mathbf{x}$ and $\mathbf{x}'$,

$$f(\mathbf{x}') \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{x}' - \mathbf{x} \rangle + \tfrac{L}{2} \|\mathbf{x}' - \mathbf{x}\|^2, \tag{39}$$

where f denotes any function with L-Lipschitz continuous gradients. Then, when $\gamma < 2/L$, we obtain

$$\mathcal{L}_t(\mathbf{x}_t^{(k+1)}) \leq \mathcal{L}_t(\mathbf{x}_t^{(k)}) - \gamma\Big(1 - \tfrac{L\gamma}{2}\Big) \|\nabla \mathcal{L}_t(\mathbf{x}_t^{(k)})\|^2. \tag{40}$$

After K iterations, we have

$$\min_{k \in [1,K]} \|\nabla \mathcal{L}_t^{(k)}\|^2 \leq \mathcal{O}\big(\tfrac{1}{K}\big), \tag{41}$$

which implies that $\mathcal{L}_t$ converges to a stationary point δ_t during the optimization of $\mathcal{P}(t)$. □

B.2 Proof of Theorem 2

Proof. Let the accumulated reconstruction error of $\mathcal{P}(t)$ be $\Gamma(t) := \|\mathcal{X}(\mathbf{x}_t) - \mathbf{x}_0\|^2$, where $\mathbf{x}_t$ denotes the estimated latent representation and $\mathbf{x}_0$ is the ground-truth image. We define $\mathcal{X}(\mathbf{x}_0) = 0$. From Theorem 1, for all $t \in \{1, \ldots, T\}$, we have:

$$\exists\, \delta_t > 0, \quad \mathcal{L}_t(\mathbf{x}_t) \leq \delta_t. \tag{42}$$

Considering Eq. (18) and Assumption 1, we obtain:

$$\Gamma(t) \leq \sum_{t=1}^{T} \left(\prod_{j=1}^{t-1} L_m^{(j)} \right) \delta_t. \tag{43}$$

Under Assumption 1, there exists a constant $L_m < 1$ such that $L_m^{(j)} \leq L_m$. Hence,

$$\Gamma(t) \leq L_m\, \Gamma(t-1) + \delta_t. \tag{44}$$

Let $\delta = \max\{\delta_1, \ldots, \delta_T\}$. By recursively applying Eq. (44) from $t = 1$ to T, we derive:

$$\begin{cases} \Gamma(1) \leq \delta, \\ \Gamma(2) \leq L_m \delta + \delta, \\ \qquad \vdots \\ \Gamma(T) \leq \delta \sum_{k=1}^{T} L_m^{k-1} = \delta \cdot \dfrac{1 - L_m^T}{1 - L_m} \leq \dfrac{\delta}{1 - L_m}. \end{cases} \tag{45}$$

Thus, $\Gamma(T)$ is bounded by $\delta/(1 - L_m)$. □

C Implementation Details

C.1 Training Details

We implement our method using PyTorch and the Hugging Face `diffusers` library. All diffusion model weights are frozen during training, and only the per-timestep noise embeddings $\boldsymbol{\epsilon}_t^*$ are optimized via gradient descent. Following *EasyInv*, we initialize the value of $\boldsymbol{\epsilon}_t^*$ by performing a single step of *DDIM Inversion*. We use the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.01. The learning rate is set to 0.1 for SD v1.4, 0.05 for both LCM-SD v1.5 and SDXL, and 0.025 for SD3. We use the DDIM sampler unless otherwise specified; for SD3, the Euler discrete sampler is adopted due to architectural compatibility. We keep the number of inversion steps equal to the number of sampling steps, denoted as T. All experiments are conducted under mixed precision: `float16` is used for SD v1.4, LCM-SD v1.5, and SDXL, while `bfloat16` is used for SD3 to ensure numerical stability. We run all experiments on a single RTX 4090 24GB GPU, except for SD3, which requires a single A100 40GB GPU due to its higher memory and compute demands.

C.2 Hyperparameters for Image Reconstruction

As described in Section 4, we tune all methods for optimal reconstruction quality. We set the number of inference steps T as follows: T=1000 for *DDIM Inversion*, T=50 for *ReNoise* and *BDIA*, T=100 for *EasyInv* and *BELM*, and T=2 for *EDICT* and our *PreciseInv*. For *ReNoise*, we disable the editing loss by setting λ=0 and sweep the renoising steps $n \in \{1, 2, \ldots, 9\}$. *EasyInv*, *EDICT*, *BDIA*, and *BELM* are used with default configurations from their respective papers. For *PreciseInv*, we set the convergence threshold η to 10^{-2}, 10^{-3} and 10^{-5}, respectively. To eliminate the effect of text conditioning, all methods are evaluated with empty prompts and guidance scale set to zero.

C.3 Pseudocode of *PreciseInv*

We present the pseudocode of the proposed *PreciseInv*. Specifically, three cases are covered: diffusion models using the DDIM sampler, diffusion models using the DDPM sampler, and rectified flow models using the Euler discrete sampler, as shown in Algorithm 2, 3, and 4, respectively.

Algorithm 2 *PreciseInv* for Diffusion Models with DDIM Sampler

Input: Real image $\mathbf{x}_0$, diffusion model $\boldsymbol{\epsilon}_\theta$, convergence threshold η, number of inference steps T
Output: Inverted latent $\mathbf{x}_T^*$
for $t = 1$ to T **do**
  Initialize $\boldsymbol{\epsilon}_t^* \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  if $t > 1$ **then**
    $\mathbf{x}_{t-1}^* \leftarrow \sqrt{\bar{\alpha}_{t-1}}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\boldsymbol{\epsilon}_{t-1}^*$
  else
    $\mathbf{x}_{t-1}^* \leftarrow \mathbf{x}_0$
  end if
  `// Apply a single DDIM Inversion step`
  $\boldsymbol{\epsilon}_t^* \leftarrow \frac{1}{\sqrt{1-\bar{\alpha}_t}}\left(\mu(\mathbf{x}_{t-1}^*) - \sqrt{\bar{\alpha}_t}\mathbf{x}_0\right)$ (μ defined in Eq. (16))
  $\mathbf{x}_t^* \leftarrow \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t^*$
  $\mathcal{L}_{\text{rec}} \leftarrow \left\|\mu(\mathbf{x}_t^*) - \mathbf{x}_{t-1}^*\right\|^2$
  while $\mathcal{L}_{\text{rec}} < \eta$ **do**
    $\mathbf{x}_t^* \leftarrow \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t^*$
    $\mathcal{L}_{\text{rec}} \leftarrow \left\|\mu(\mathbf{x}_t^*) - \mathbf{x}_{t-1}^*\right\|^2$
    $\boldsymbol{\epsilon}_t^* \leftarrow \boldsymbol{\epsilon}_t^* - \nabla\mathcal{L}_{\text{rec}}$
  end while
end for
return $\mathbf{x}_T^*$

Algorithm 3 *PreciseInv* for Diffusion Models with DDPM Sampler

Input: Real image $\mathbf{x}_0$, diffusion model $\boldsymbol{\epsilon}_\theta$, convergence threshold η, number of inference steps T
Output: Inverted latent $\mathbf{x}_T^*$, additional noise terms $\{\mathbf{z}_T^*, \mathbf{z}_{T-1}^*, \cdots, \mathbf{z}_1^*\}$
for $t = 1$ to T **do**
  Initialize $\boldsymbol{\epsilon}_t^* \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Initialize $\mathbf{z}_t^* \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  if $t > 1$ **then**
    $\mathbf{x}_{t-1}^* \leftarrow \sqrt{\bar{\alpha}_{t-1}}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\boldsymbol{\epsilon}_{t-1}^*$
  else
    $\mathbf{x}_{t-1}^* \leftarrow \mathbf{x}_0$
  end if
  $\boldsymbol{\epsilon}_t^* \leftarrow \frac{1}{\sqrt{1-\bar{\alpha}_t}}\left(\mu(\mathbf{x}_{t-1}^*) - \sqrt{\bar{\alpha}_t}\mathbf{x}_0\right)$ (μ defined in Eq. (24))
  $\mathbf{x}_t^* \leftarrow \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t^*$
  $\mathcal{L}_{\text{rec}} \leftarrow \left\|\mu(\mathbf{x}_t^*, \mathbf{z}_t^*) - \mathbf{x}_{t-1}^*\right\|^2$
  while $\mathcal{L}_{\text{rec}} < \eta$ **do**
    $\mathbf{x}_t^* \leftarrow \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t^*$
    $\mathcal{L}_{\text{rec}} \leftarrow \left\|\mu(\mathbf{x}_t^*, \mathbf{z}_t^*) - \mathbf{x}_{t-1}^*\right\|^2$
    $\boldsymbol{\epsilon}_t^* \leftarrow \boldsymbol{\epsilon}_t^* - \nabla\mathcal{L}_{\text{rec}}$
    $\mathbf{z}_t^* \leftarrow \mathbf{z}_t^* - \nabla\mathcal{L}_{\text{rec}}$
  end while
end for
return $\mathbf{x}_T^*$, $\{\mathbf{z}_T^*, \mathbf{z}_{T-1}^*, \cdots, \mathbf{z}_1^*\}$

D Additional Results

D.1 Quantitative Results.

We provide additional quantitative results of our method on the image reconstruction task. Table 4 presents the performance of *PreciseInv* with different combinations of base models and samplers, as described in Section A. The experiments run on the COCO 2017 validation set [24]. We set the number of inference steps to $T = 3$ for SD3 and $T = 2$ for the other models. The convergence threshold $\eta = 10^{-3}$ is used for all models.

Algorithm 4 *PreciseInv* for Rectified Flow with Euler Discrete Sampler

Input: Real image $\mathbf{x}_0$, rectified flow model $\boldsymbol{\epsilon}_\theta$, convergence threshold η, number of inference steps T, scalars s, m
Output: Inverted latent $\mathbf{x}_T^*$
`// Initialize discrete timesteps`
for $i = 1$ to T **do**
  $t_i \leftarrow \pi_{\text{ln}}(i; s, m)$
end for
for $i = 1$ to T **do**
  Initialize $\boldsymbol{\epsilon}_{t_i}^* \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  if $t > 1$ **then**
    $\mathbf{x}_{t_{i-1}}^* \leftarrow (1 - t_{i-1})\,\mathbf{x}_0 + t_{i-1}\,\boldsymbol{\epsilon}_{t_{i-1}}^*$
  else
    $\mathbf{x}_{t_{i-1}}^* \leftarrow \mathbf{x}_0$
  end if
  $\mathbf{x}_{t_i}^* \leftarrow (1 - t_i)\,\mathbf{x}_0 + t_i\,\boldsymbol{\epsilon}_{t_i}^*$
  $\mathcal{L}_{\text{rec}} \leftarrow \|\mu(\mathbf{x}_{t_i}^*, t_i, t_{i-1}) - \mathbf{x}_{t_{i-1}}^*\|^2$ (μ defined in Eq. (36))
  while $\mathcal{L}_{\text{rec}} > \eta$ **do**
    $\mathbf{x}_{t_i}^* \leftarrow (1 - t_i)\,\mathbf{x}_0 + t_i\,\boldsymbol{\epsilon}_{t_i}^*$
    $\mathcal{L}_{\text{rec}} \leftarrow \|\mu(\mathbf{x}_{t_i}^*, t_i, t_{i-1}) - \mathbf{x}_{t_{i-1}}^*\|^2$
    $\boldsymbol{\epsilon}_{t_i}^* \leftarrow \boldsymbol{\epsilon}_{t_i}^* - \nabla\mathcal{L}_{\text{rec}}$
  end while
end for
$\mathbf{x}_{t_T}^* \leftarrow (1 - t_T)\,\mathbf{x}_0 + t_T\,\boldsymbol{\epsilon}_{t_T}^*$
return $\mathbf{x}_T^*$

Table 4 shows that all models achieve comparable reconstruction quality. SD3 attains higher SSIM and PSNR scores, while other models perform better on LPIPS. Compared to the results reported in Tables 1 and 2, *PreciseInv* maintains state-of-the-art reconstruction performance, further demonstrating the generality and robustness of our method. However, SD3 exhibits lower inference efficiency relative to other models, which can be attributed to its larger model size and the absence of an effective initialization manner.

To address this, future work will explore a progressively decreasing timestep optimization strategy for large models such as SD3. We believe that this strategy will result in a smoother optimization landscape. Additionally, exploring better initialization techniques, such as using initial values derived from existing flow-based inversion methods, would also be a promising direction.

Table 4: Additional quantitative results for image reconstruction, including: (i) Stable Diffusion 3.5 medium model with default Euler Discrete sampler, and (ii) Stable Diffusion v1.4, Latent Consistent Model Stable Diffusion v1.5, and Stable Diffusion XL with DDPM sampler.

Model	Sampler	LPIPS (↓)	SSIM (↑)	PSNR (↑)	Time (s, ↓)
SD3	Euler	0.104	0.896	30.12	403.31
SD v1.4	DDPM	0.080	0.764	25.93	9.08
LCM-SD v1.5	DDPM	0.076	0.784	26.84	12.07
SDXL	DDPM	0.091	0.842	30.39	18.59

We also report reconstruction performance under varying convergence thresholds η and inference step counts T in Table 5. We randomly sample 20 images from the COCO 2017 validation set and evaluate using the Stable Diffusion v1.4 model with a DDIM sampler. The number of inference steps is varied as $T \in 2, 5, 10, 20, 50, 100$ and the convergence threshold as $\eta \in 10^{-2}, 10^{-3}, 10^{-5}, 10^{-6}, 10^{-7}$. We report metrics including LPIPS, SSIM, and PSNR, as well as inference time in seconds.

From Table 5, we draw the following observations. First, for a fixed number of inference steps T, decreasing the convergence threshold η consistently improves reconstruction quality across all metrics—LPIPS decreases, while SSIM and PSNR increase. However, this comes at the cost of

significantly longer inference time, especially as η drops below 10^{-5}. Second, we observe diminishing returns in reconstruction quality when $\eta < 10^{-5}$, particularly for small-step regimes (e.g., $T \leq 10$). This suggests that extremely strict convergence criteria may be unnecessary when the number of inference steps is limited. Third, when η is fixed, increasing the number of inference steps T initially degrades the reconstruction quality before improving it. This phenomenon results from a trade-off: while smaller per-step initialization errors benefit from higher T, the total accumulated error at convergence (approximately $T \cdot \eta$) also increases. Notably, due to the smooth structure of the diffusion latent space, error accumulation is sublinear and does not dominate overall performance. Fourth, inference time exhibits a non-monotonic trend with respect to T. When $\eta \geq 10^{-5}$, increasing T initially reduces inference time—likely due to faster convergence from smaller per-step errors—but then increases it again as the step count becomes the dominant factor. In contrast, for stricter thresholds ($\eta \leq 10^{-5}$), inference time generally decreases with larger T, implying that the time saved through easier optimization outweighs the overhead introduced by additional steps. Finally, when $T \leq 10$, tightening the threshold from $\eta = 10^{-5}$ to $\eta = 10^{-6}$ yields negligible gains in reconstruction quality. Moreover, for large-step configurations ($T > 10$), using $\eta = 10^{-7}$ achieves reconstruction performance on par with small-step setups at $\eta = 10^{-5}$. Taken together, these findings suggest that setting $\eta = 10^{-5}$ and $T = 2$ strikes a favorable balance between efficiency and quality.

Table 5: Image reconstruction results under varying convergence thresholds η and inference steps T. The table reports four metrics: LPIPS (↓), SSIM (↑), PSNR (↑), and inference time in seconds (↓).

Metric	η	$T=2$	$T=5$	$T=10$	$T=20$	$T=50$	$T=100$
LPIPS	10^{-2}	0.069	0.205	0.138	0.095	0.044	0.036
	10^{-3}	0.037	0.052	0.139	0.096	0.044	0.036
	10^{-5}	0.031	0.031	0.034	0.074	0.044	0.036
	10^{-6}	0.031	0.031	0.031	0.037	0.045	0.036
	10^{-7}	-	-	-	0.031	0.036	0.032
SSIM	10^{-2}	0.876	0.839	0.864	0.877	0.894	0.896
	10^{-3}	0.894	0.886	0.867	0.877	0.894	0.896
	10^{-5}	0.897	0.898	0.896	0.059	0.895	0.896
	10^{-6}	0.897	0.898	0.898	0.896	0.895	0.897
	10^{-7}	-	-	-	0.898	0.897	0.898
PSNR	10^{-2}	31.72	26.75	26.42	26.91	31.30	32.65
	10^{-3}	33.12	29.75	26.15	26.89	31.30	32.65
	10^{-5}	33.33	33.33	32.76	28.04	31.10	32.63
	10^{-6}	33.33	33.35	33.37	33.03	30.62	32.63
	10^{-7}	-	-	-	33.33	33.11	33.29
Time (s)	10^{-2}	4.05	2.27	1.50	2.42	5.18	9.56
	10^{-3}	8.25	7.13	2.61	3.85	5.08	9.82
	10^{-5}	23.76	24.82	37.77	21.44	7.16	12.23
	10^{-6}	99.60	82.06	55.77	68.18	62.99	26.68
	10^{-7}	-	-	-	242.63	160.51	107.98

D.2 Qualitative Results.

Here, we present additional qualitative results on image reconstruction, prompt-based image editing, and image interpolation.

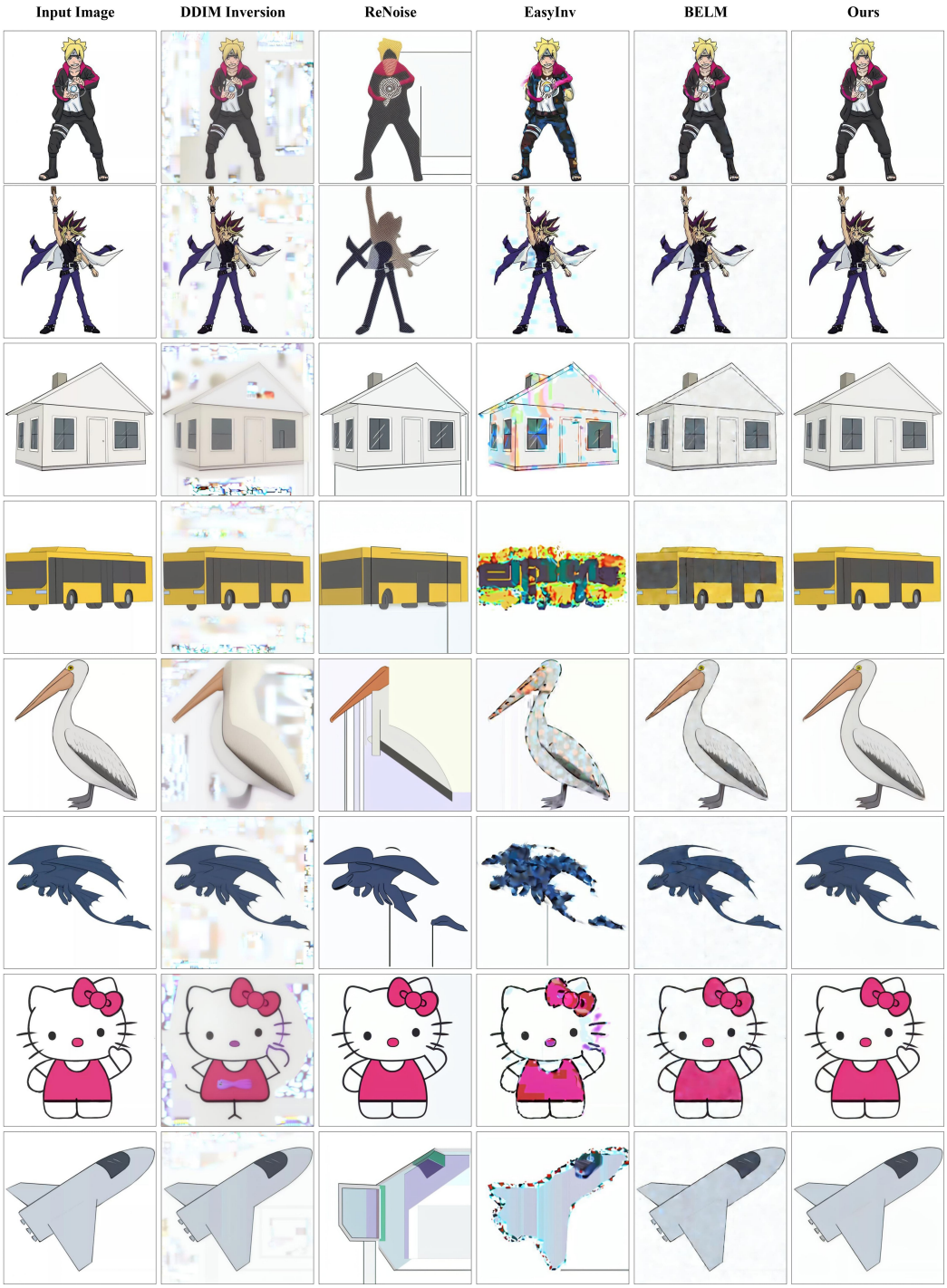

Figure 6: Qualitative comparison of image reconstruction results from different diffusion inversion methods using Stable Diffusion v1.4 on the Cartoon dataset.

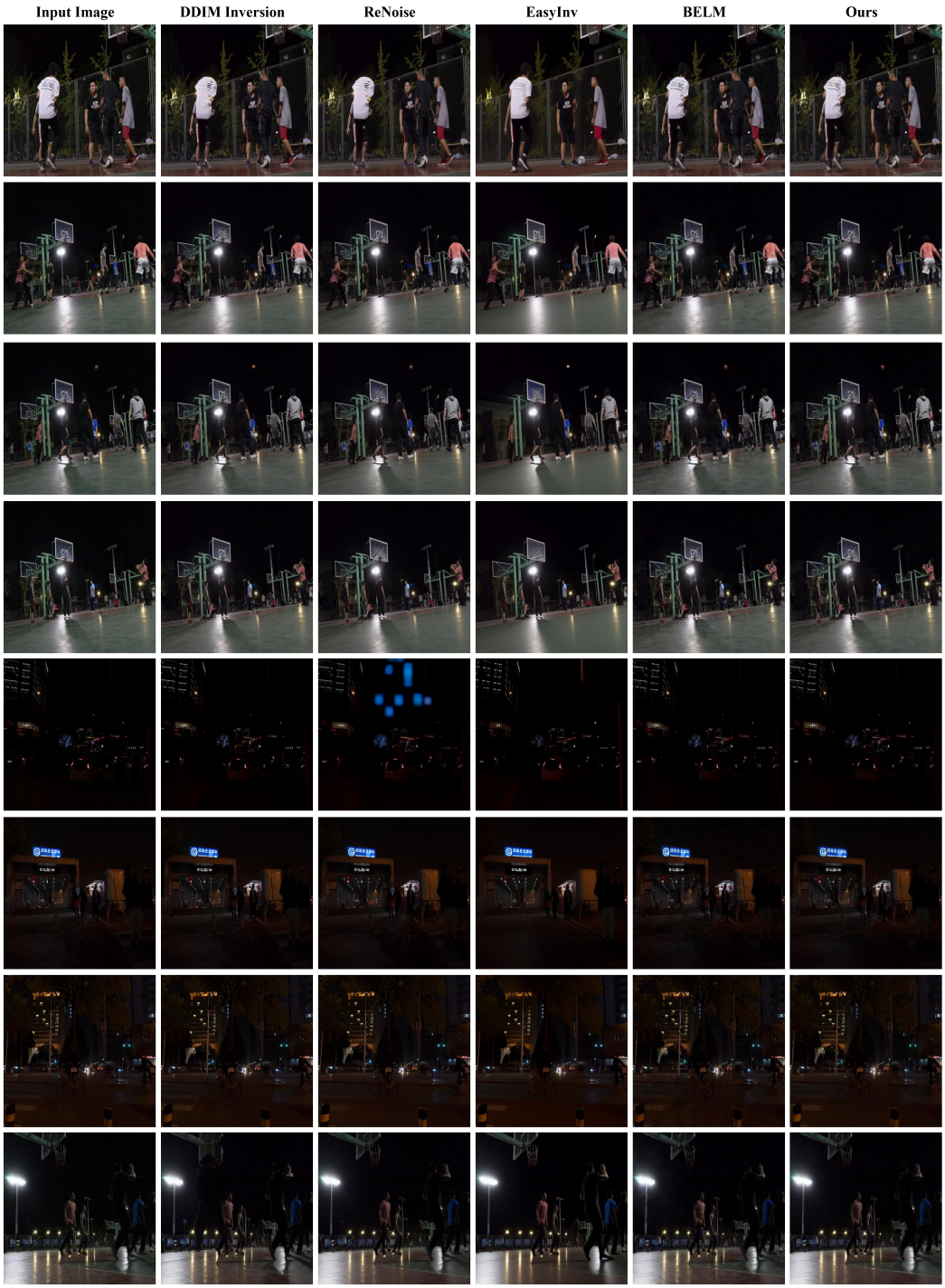

Figure 7: Qualitative comparison of image reconstruction results from different diffusion inversion methods using Stable Diffusion v1.4 on the DarkFace dataset.

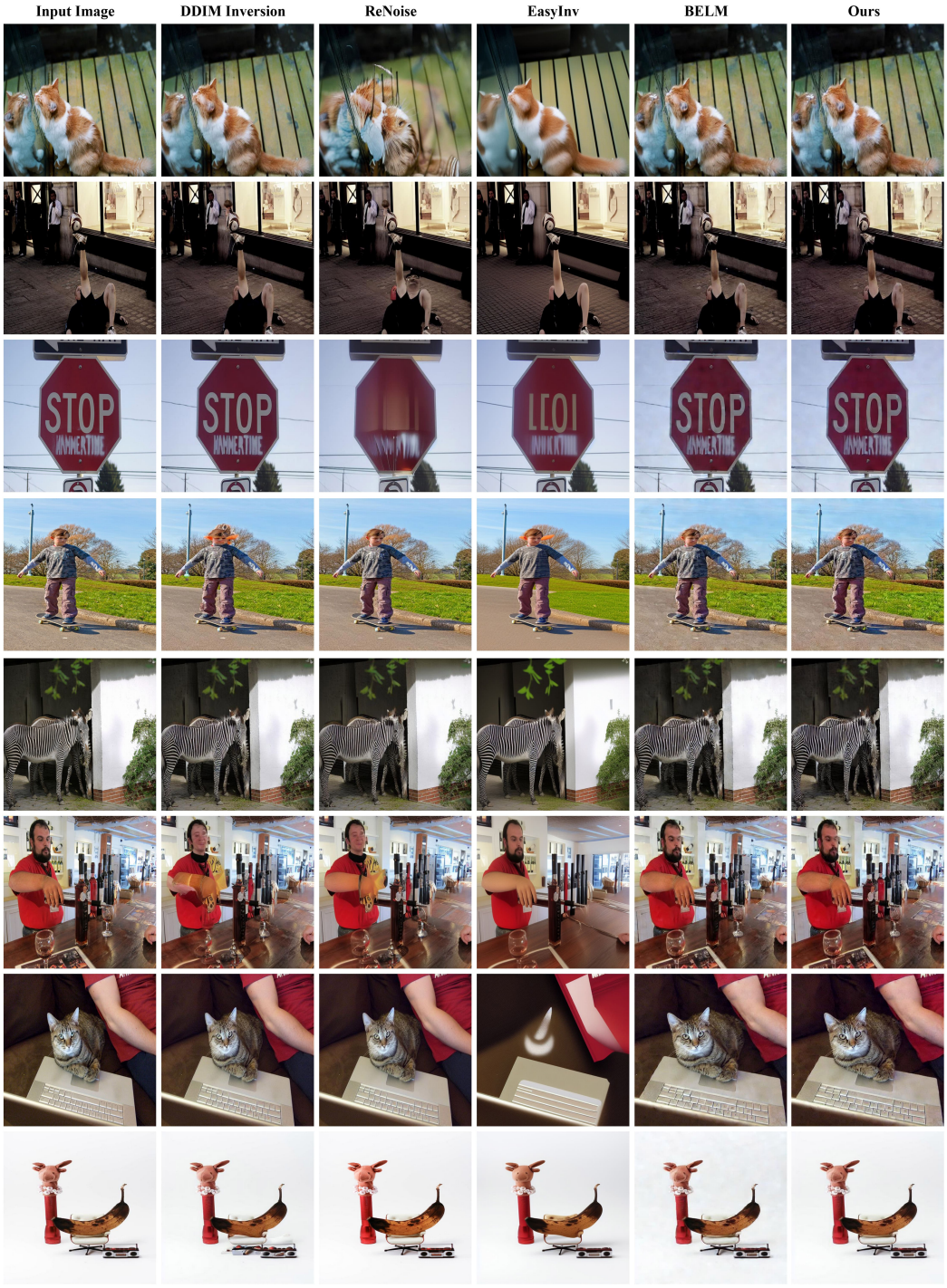

Figure 8: Qualitative comparison of image reconstruction results from different diffusion inversion methods using Stable Diffusion v1.4 on the COCO dataset.

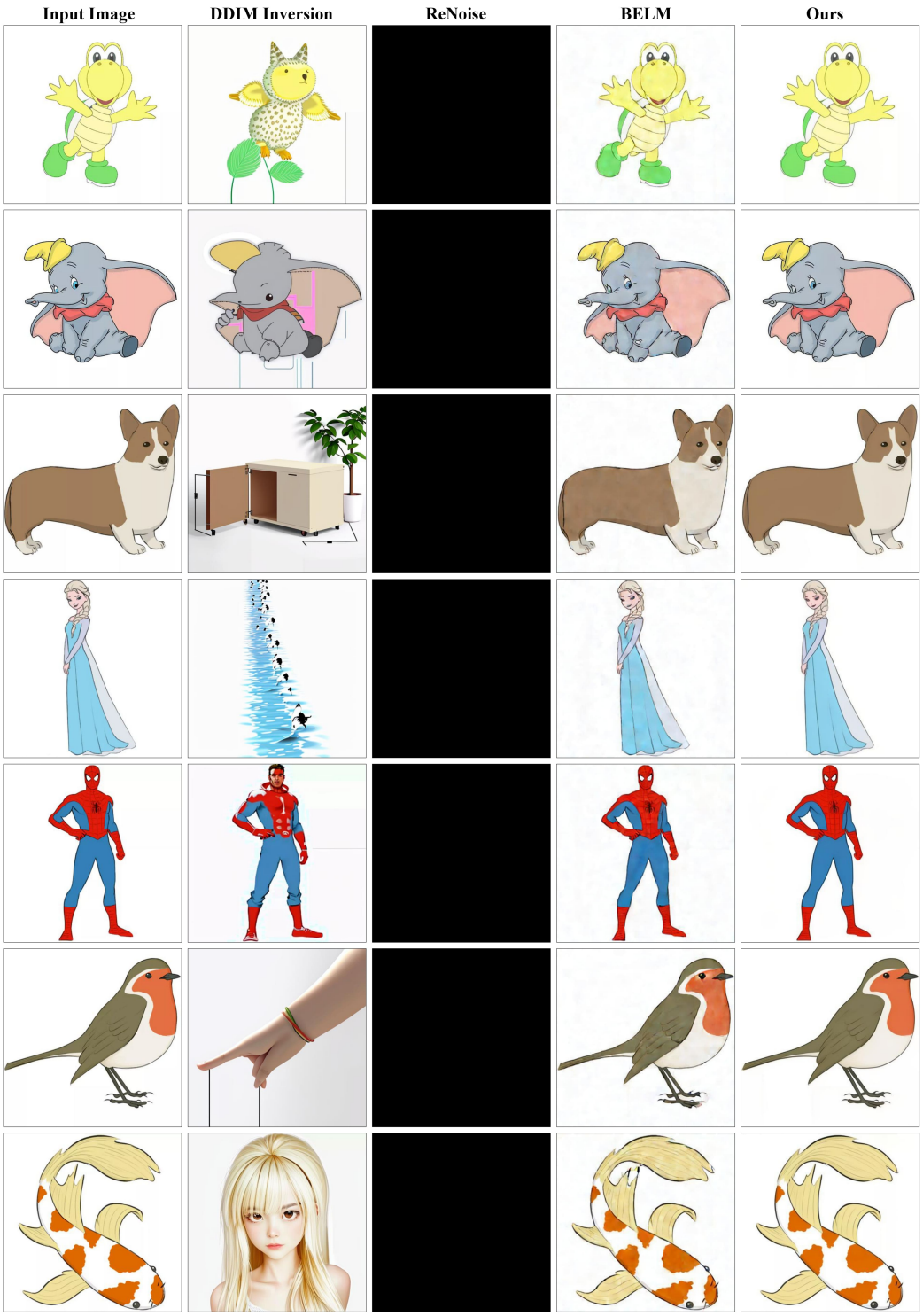

Figure 9: Qualitative comparison of image reconstruction results from different diffusion inversion methods using the Latent Consistency Model with Stable Diffusion v1.5 on the Cartoon dataset.

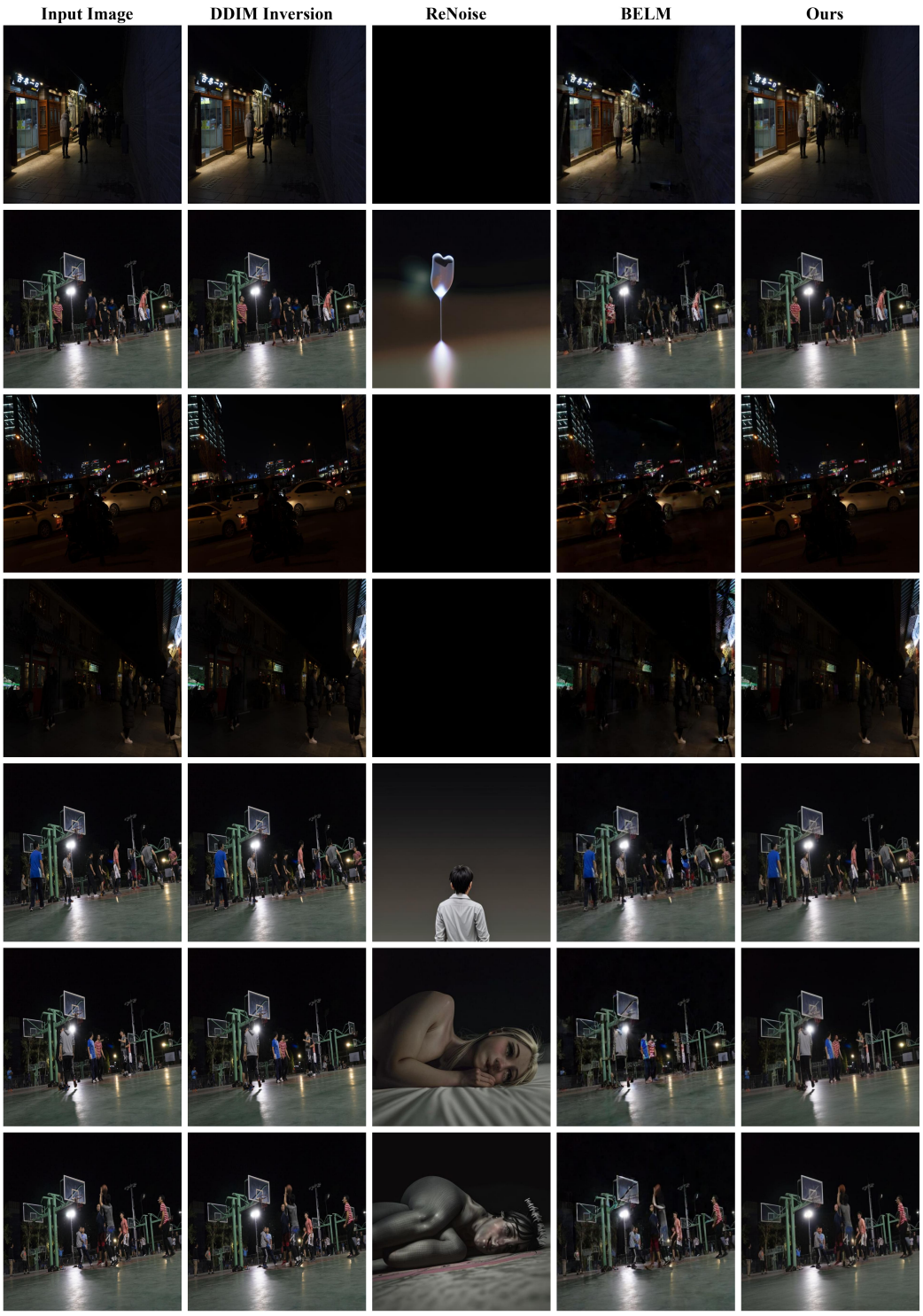

Figure 10: Qualitative comparison of image reconstruction results from different diffusion inversion methods using the Latent Consistency Model with Stable Diffusion v1.5 on the DarkFace dataset.

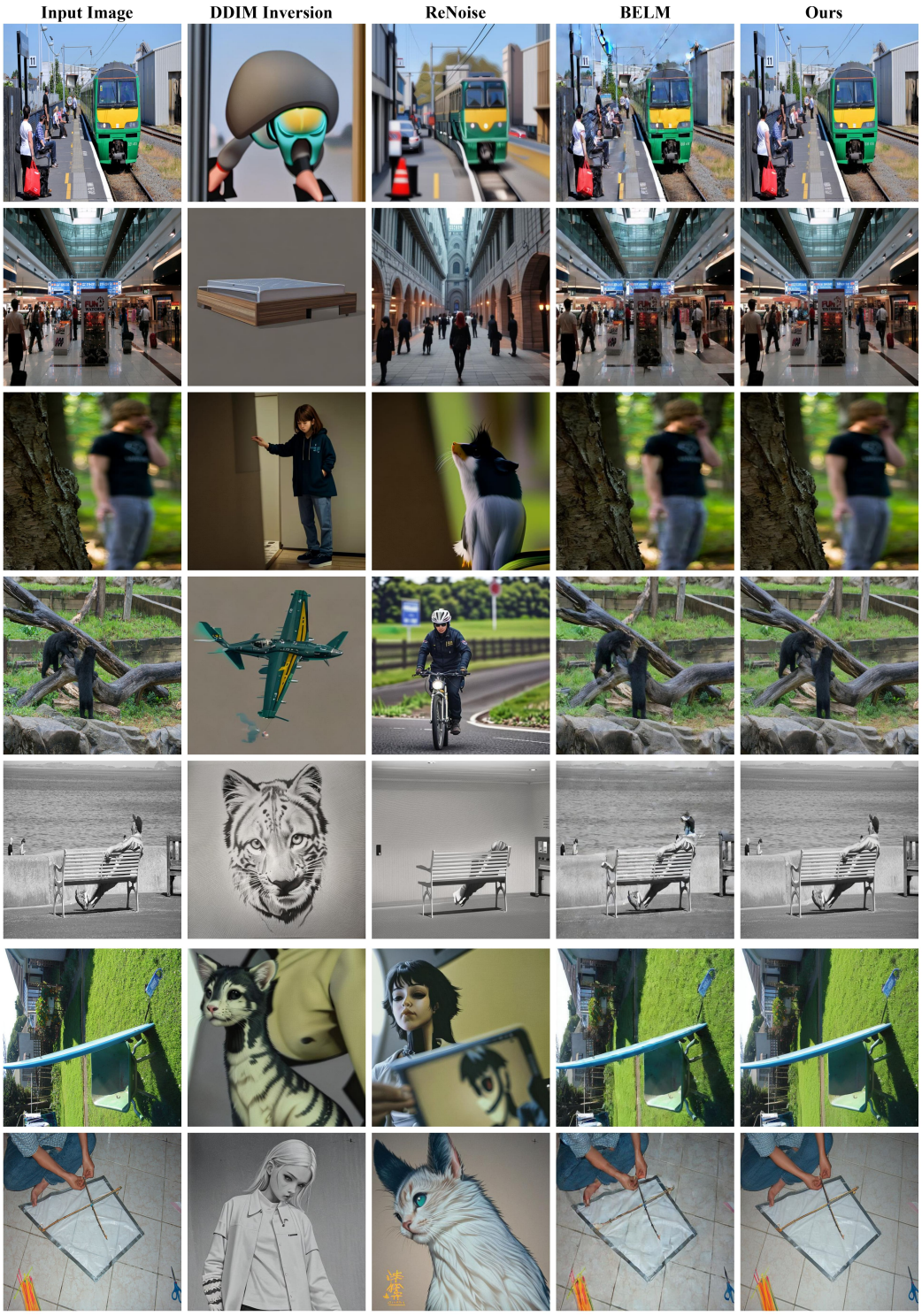

Figure 11: Qualitative comparison of image reconstruction results from different diffusion inversion methods using the Latent Consistency Model with Stable Diffusion v1.5 on the COCO dataset.

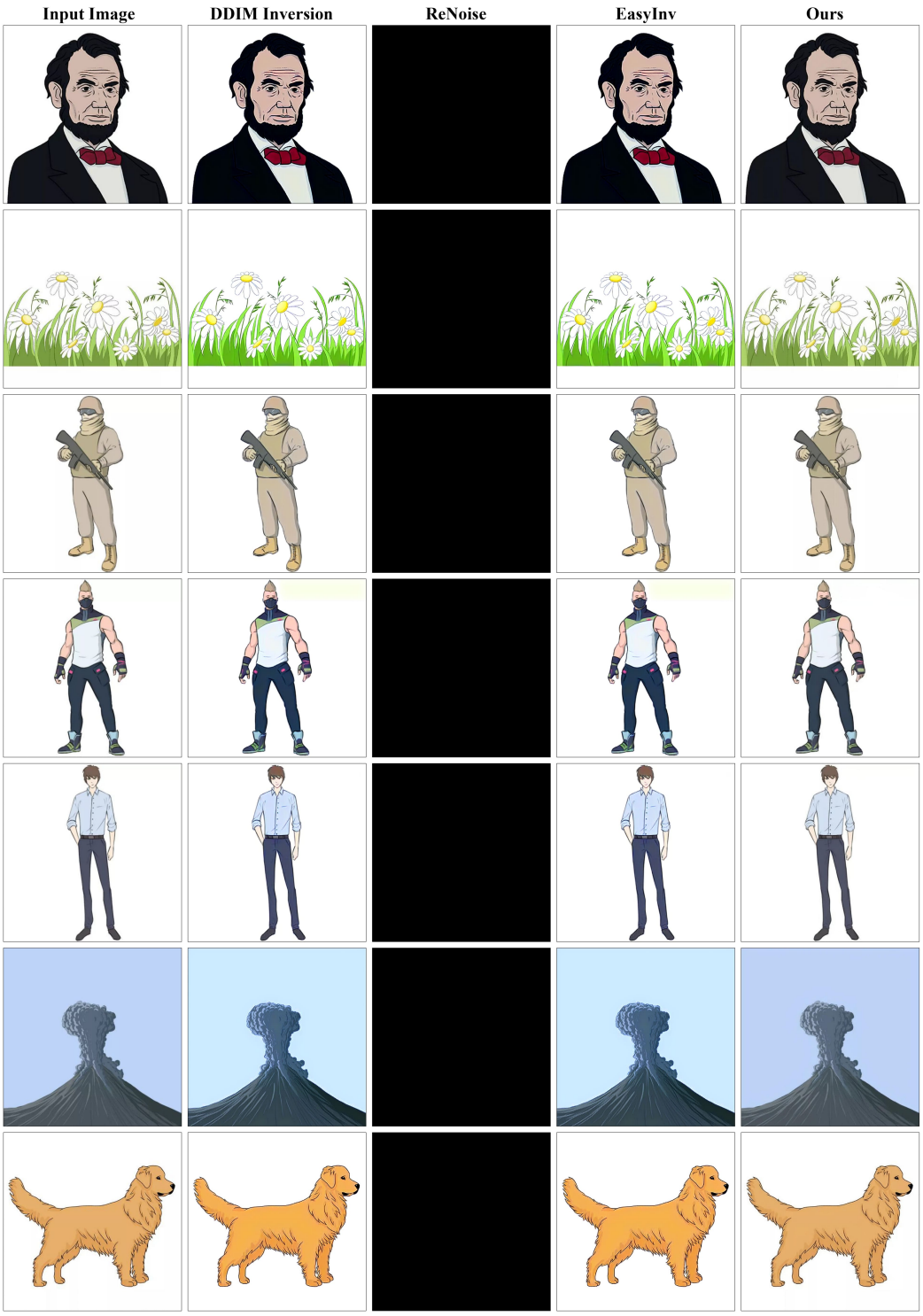

Figure 12: Qualitative comparison of image reconstruction results from different diffusion inversion methods using the Stable Diffusion XL on the Cartoon dataset.

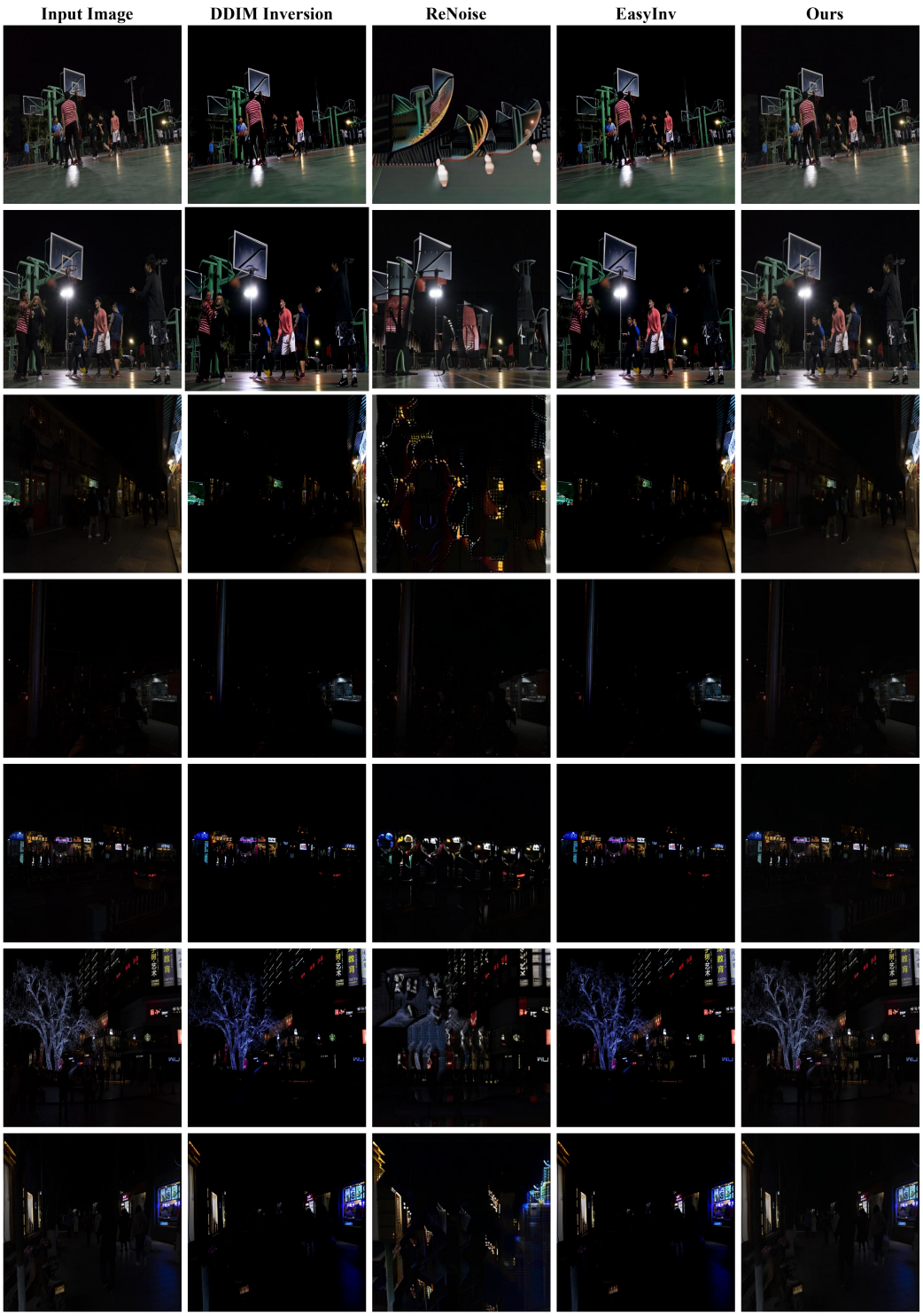

Figure 13: Qualitative comparison of image reconstruction results from different diffusion inversion methods using the Stable Diffusion XL on the DarkFace dataset.

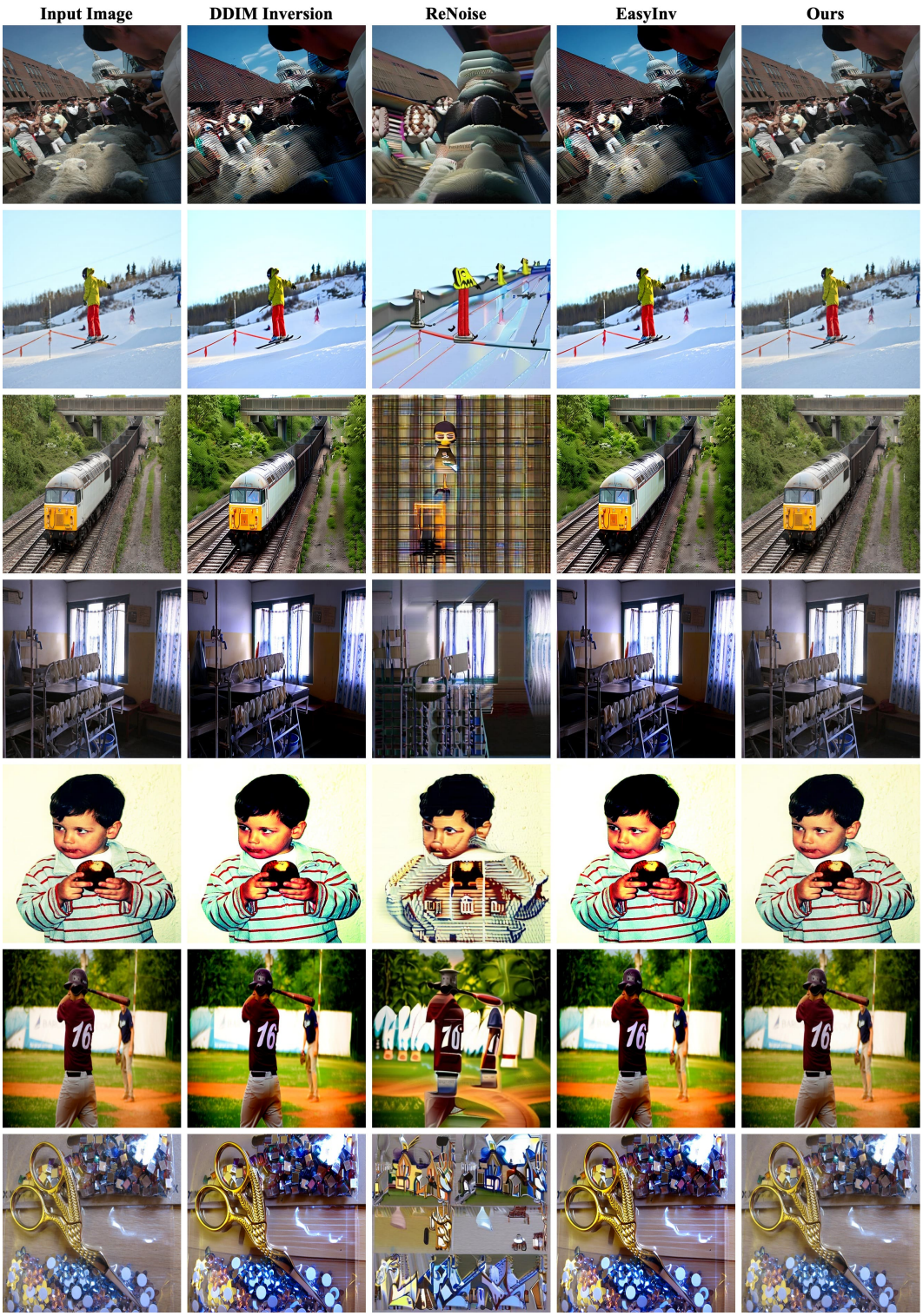

Figure 14: Qualitative comparison of image reconstruction results from different diffusion inversion methods using the Stable Diffusion XL on the COCO dataset.

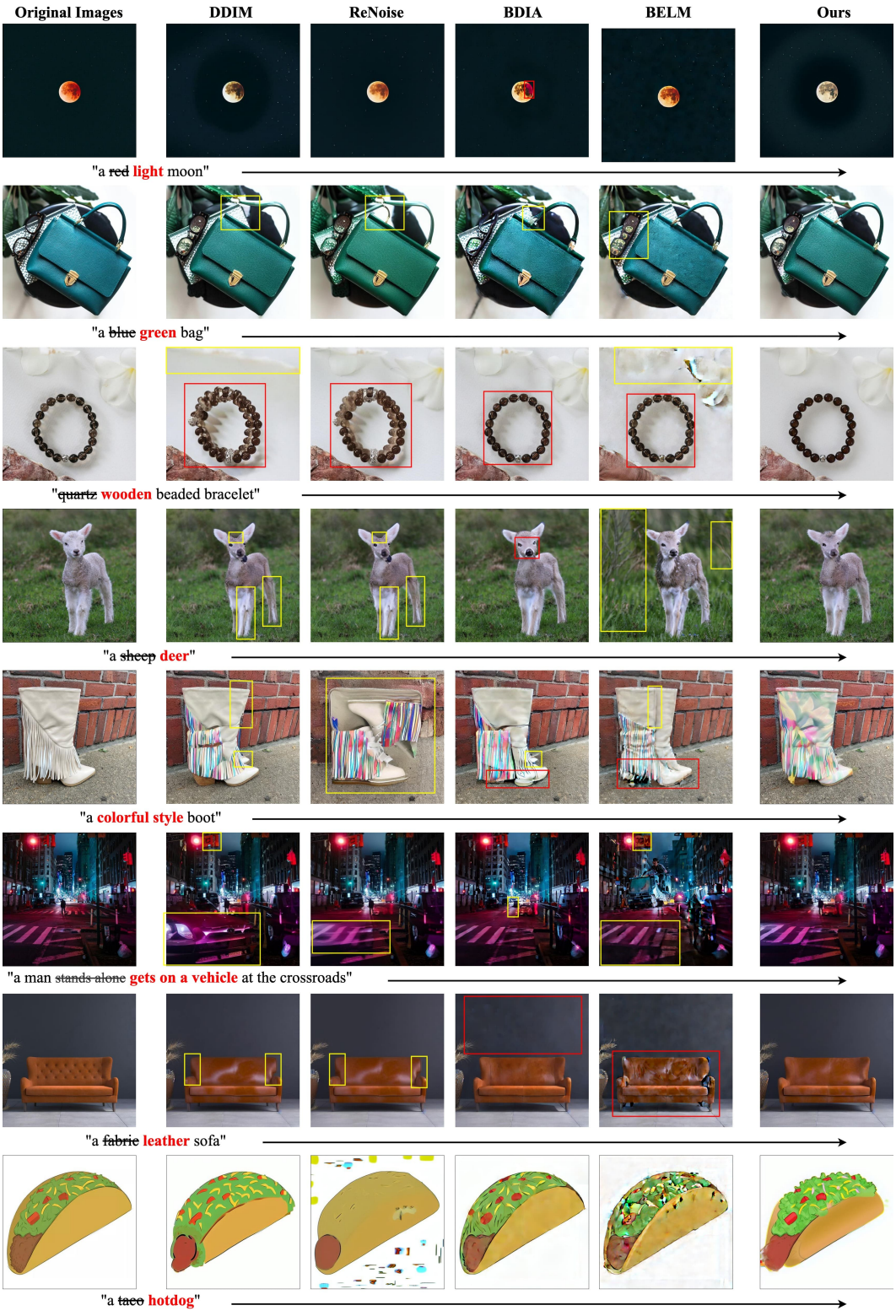

Figure 15: Qualitative comparison of prompt-driven image editing results from different diffusion inversion methods using the Stable Diffusion v1.4.

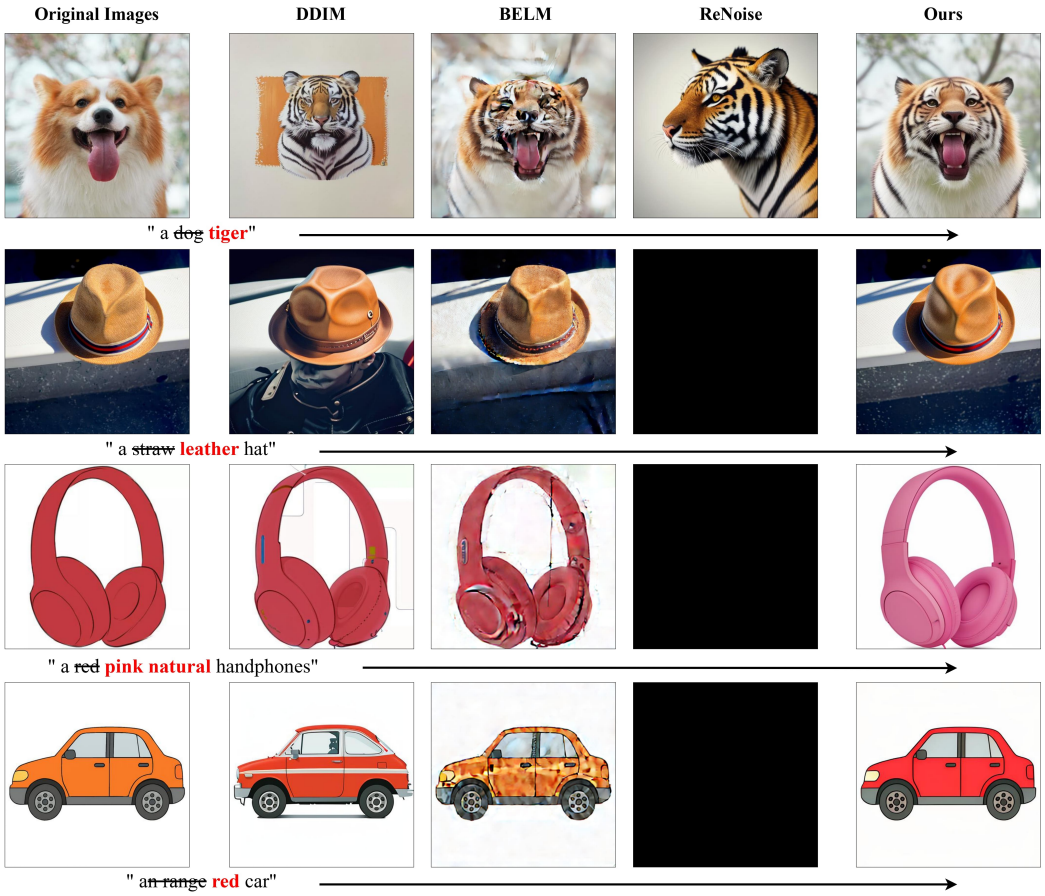

Figure 16: Qualitative comparison of prompt-driven image editing results from different diffusion inversion methods using the Latent Consistency Model with Stable Diffusion v1.5.

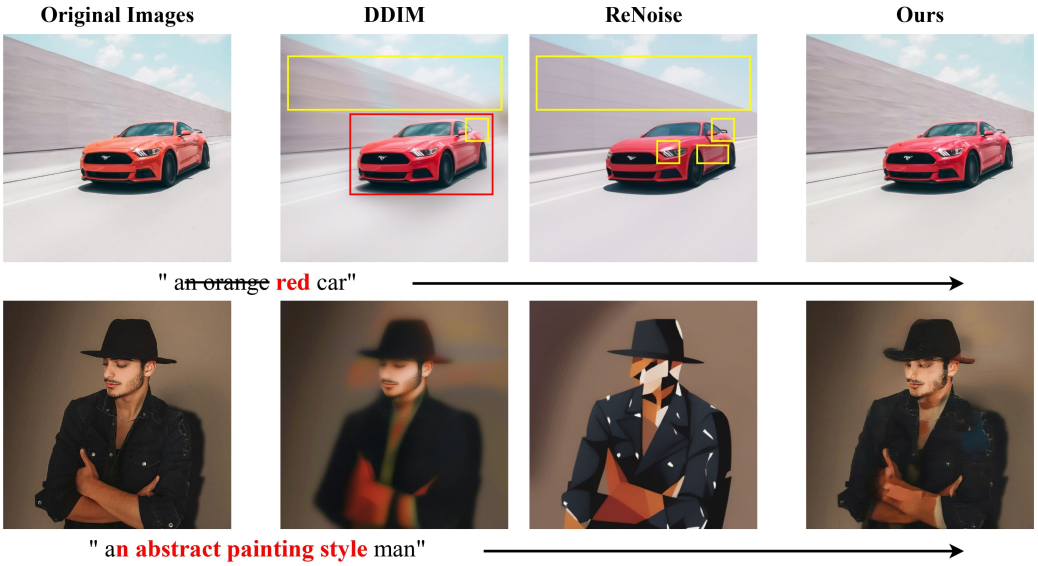

Figure 17: Qualitative comparison of prompt-driven image editing results from different diffusion inversion methods using the Stable Diffusion SDXL.

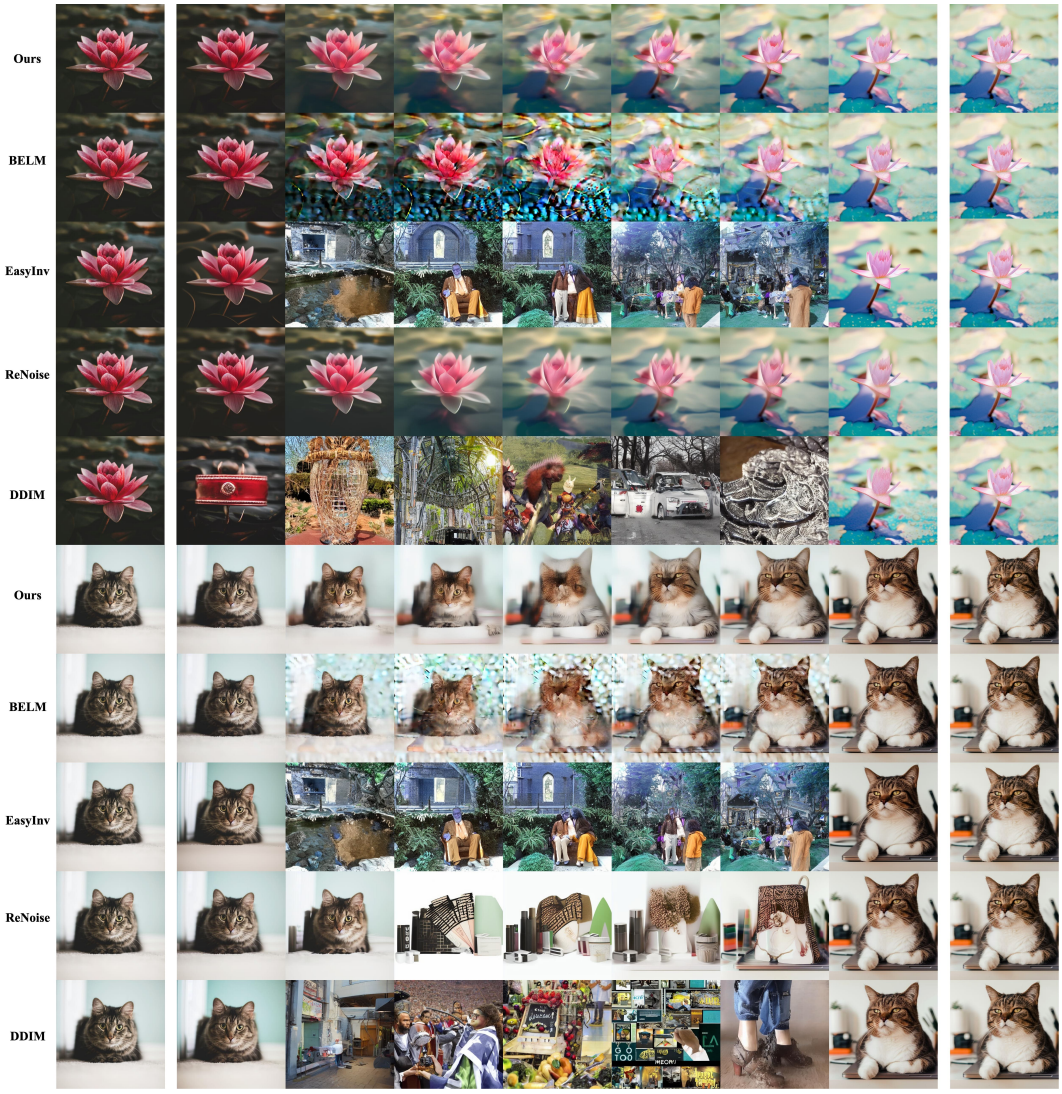

Figure 18: Qualitative comparison of image interpolation from different diffusion inversion methods using the Stable Diffusion v1.4.

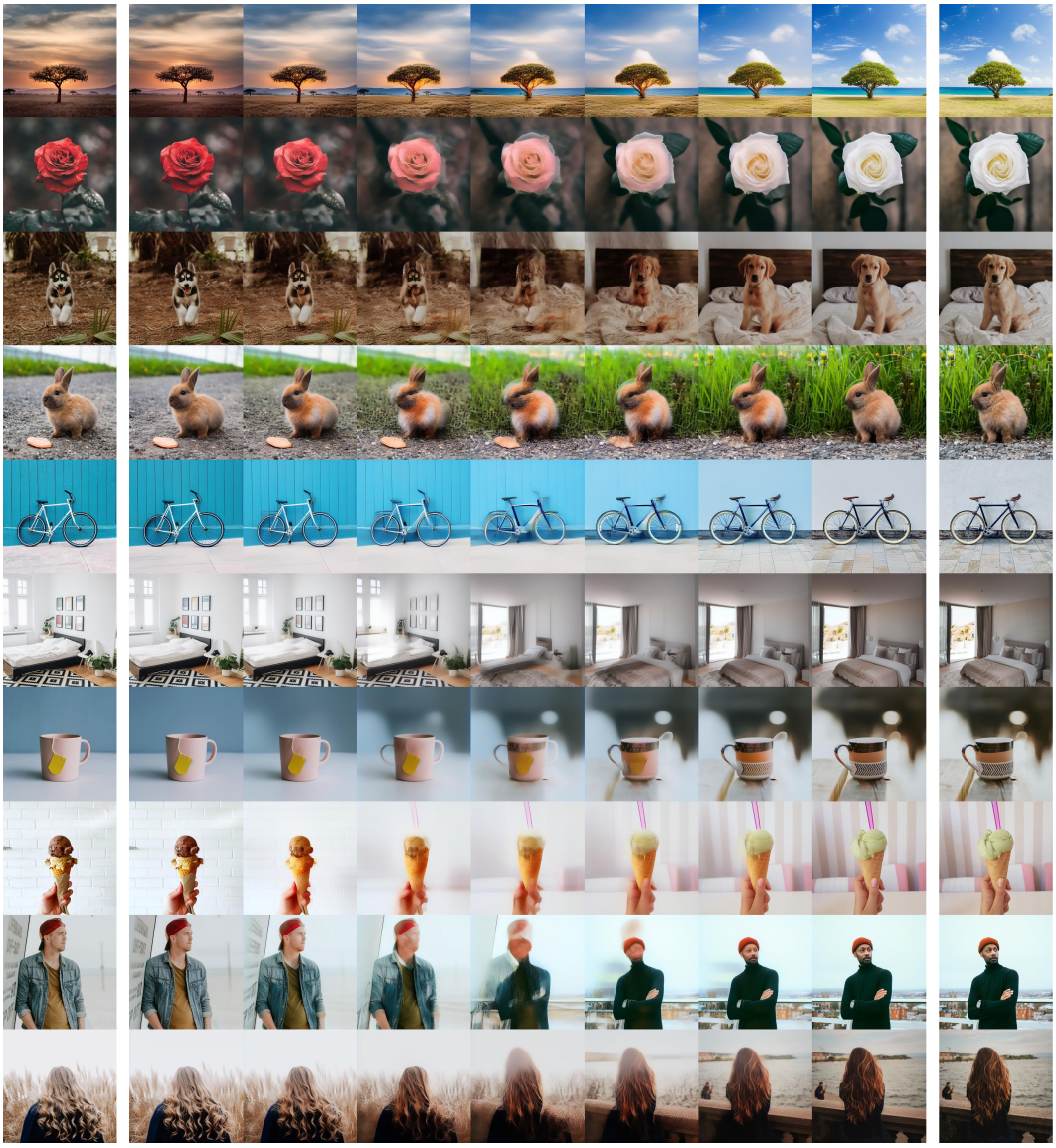

Figure 19: Additional qualitative result of image interpolation from the proposed *PreciseInv* using the Stable Diffusion v1.4.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the claims, and the experimental results well support the claims.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: See Appendix.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: See Section 3.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide all the necessary information to reproduce the main experimental results.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We will release the source code.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: See Section 3.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: We follow similar recent work on diffusion inversion that doesn't include statistical significance of the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: This paper provides sufficient information on the computer resources.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We strictly adhere to the NeurIPS ethical NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: See Section 4.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite the original paper that provides the models or datasets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

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