AutoEdit: Automatic Hyperparameter Tuning for Image Editing

Chau Pham[1] **Quan Dao**[2] **Mahesh Bhosale**[1] **Yunjie Tian**[1*]

Dimitris Metaxas[2] **David Doermann**[1]

[1]University at Buffalo
[2]Rutgers University

Abstract

Recent advances in diffusion models have revolutionized text-guided image editing, yet existing editing methods face critical challenges in hyperparameter identification. To get the reasonable editing performance, these methods often require the user to brute-force tune multiple interdependent hyperparameters, such as inversion timesteps and attention modification, *etc.* This process incurs high computational costs due to the huge hyperparameter search space. We consider searching optimal editing's hyperparameters as a sequential decision-making task within the diffusion denoising process. Specifically, we propose a reinforcement learning framework, which establishes a Markov Decision Process that dynamically adjusts hyperparameters across denoising steps, integrating editing objectives into a reward function. The method achieves time efficiency through proximal policy optimization while maintaining optimal hyperparameter configurations. Experiments demonstrate significant reduction in search time and computational overhead compared to existing brute-force approaches, advancing the practical deployment of a diffusion-based image editing framework in the real world. Codes can be found at `https://github.com/chaupham1709/AutoEdit.git`.

1 Introduction

Image generation has recently witnessed remarkable advancements through diffusion models [13, 34, 3, 32, 36], driving a growing interest in their broad applicability. Within this domain, prompt-to-prompt image editing [12, 5, 4, 16, 45, 26, 14, 8] has emerged as a critical subfield focused on modifying image content according to textual instructions. This task focuses on achieving two essential objectives: (1) precisely aligning the modified image with the editing prompt, while (2) minimizing unnecessary alterations to the original content beyond those required by the editing instruction. Maintaining this delicate balance between instruction alignment and background preservation is the primary challenge.

Despite notable advancements, current image editing methods [5, 12, 39, 15, 16, 17] suffer from a heavy reliance on manual hyperparameter identification during the editing process. These hyperparameters span multiple dimensions: inversion timestep configuration [39, 15, 11], attention modulation mechanisms [12, 16] (including layer-wise modification and semantic reweighting), and feature blending coefficients [17, 11]. As empirically demonstrated in [45], the optimal hyperparameter combination exhibits strong sensitivity across different images. This requires exhaustive trial-and-error iterations to identify optimal hyperparameter sets that simultaneously satisfy two

*Corresponding author

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

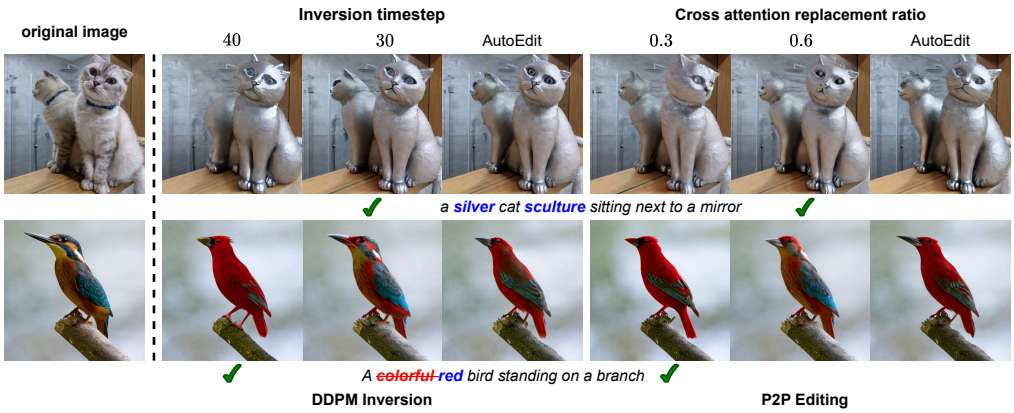


Figure 1: **Optimal hyperparameters vary significantly across images**: The cat image achieves best editing at step 30 while the bird requires step 40 in timestep experiments, with similar variance observed in P2P [12] cross-attention ratios. Our AutoEdit automatically identifies near-optimal configurations across these parameters, matching manual search performance (last column).

criteria: (1) achieving instruction-aligned editing results while (2) preserving original background. Such a hyperparameter search introduces significant computational overhead and creates substantial usability barriers for non-expert users, as evidenced by our quantitative illustration in Figure 1.

Previous work OIR [45] optimize the value of hyperparameter "inversion timestep r" , which is the maximum timestep we try to invert image in editing process, for each pair (input images, edit prompt) to obtain better editing result. Specifically, OIR proposes a brute-force search strategy that evaluates all possible $r \in \{1, 2, \ldots, T\}$ (T is total denoising step) and selects the optimal r^* to achieve the highest editing score. Furthermore, this approach requires T denoising processes per editing sample, which consume $\mathcal{O}(T^2)$ number of functional evaluation (NFEs). This method becomes impractical when T is large, which is around 50 or 100 in case of editing process. However, OIR only focuses on choosing hyperparameter r while neglecting other critical hyperparameters.

For each editing method, we define a hyperparameter set $\mathcal{H} = \{h^1, h^2, ..., h^K\}$, consisting of K distinct hyperparameters. Assuming each h^k can take on N possible values, a brute-force search for the optimal configuration incurs a computational complexity of $\mathcal{O}(TN^K)$, which is impractical in real-world scenarios. To address this, we propose an efficient hyperparameter identification framework that reduces the search complexity from exponential $\mathcal{O}(TN^K)$ to linear $\mathcal{O}(T)$. Conceptually, a diffusion-based editing process can be viewed as a denoising procedure influenced by the hyperparameter set $\mathcal{H}$. This perspective allows us to formulate the editing process as RL-driven Markov Decision Process (MDP). Specifically, each denoising step $t = T \rightarrow 1$ corresponds to a state $s_t = (x_t, t)$, where x_t is the noisy latent at timestep t. We treat each hyperparameter h^k as a sequence $\{h_t^k\}_{t=T\rightarrow 1}$, and collectively, $\mathcal{H}_t = \{h_t^k\}_{k=1}^K$ represents a set of parallel actions taken at state s_t. The reward function is designed to integrate two essential editing criteria: prompt alignment and background preservation. Optimization is performed using Proximal Policy Optimization (PPO) [37]. At each timestep t, the learned policy selects the optimal hyperparameter set $\mathcal{H}_t$, which is directly applied during editing process. As the result, our technique could save the user from heuristically choosing hyperparamenter.

Our core contributions are threefold: (1) First formulating optimal hyperparameter identification as a key challenge in diffusion-based editing, exposing the computational bottleneck $\mathcal{O}(TN^k)$ of brute force search; (2) A reinforcement learning framework that unifies hyperparameter identification with standard denoising, achieving efficient $\mathcal{O}(T)$ optimal hyperparameter search; (3) Empirical validation showing that our method nearly achieves optimal hyperparameter while reducing search time by around three times versus brute-force baselines.

2 Related work

Text-prompted Image Editing. Text-based image editing [4, 5, 12, 8, 11, 40, 38, 31] modifies image content based on editing instructions, aiming to generate edited images that faithfully adhere to the

editing prompts while preserving most original background from the source image. While Generative Adversarial Networks (GANs) [27, 21, 43] achieved partial success in domain-specific datasets (e.g., facial images), they struggle with text-based editing due to low performance of text-to-image GAN. This limitation has been substantially addressed by pretrained text-to-image diffusion models [34], which have enabled the emergence of sophisticated editing frameworks [4, 5, 12, 8, 11, 40, 38, 31] through three principal paradigms: (1) prompt-to-prompt manipulation [5, 12, 15, 16, 17, 45], (2) instruction-based editing [4, 14, 31], and (3) image personalization [35, 9]. However, existing methods require brute-force search for optimal editing hyperparameters, which is a non-trivial process due to extensive trial-and-error iterations. Motivated from reinforcement learning in improving image generation [24, 10, 46, 29], we propose an RL framework for searching optimal hyperparameter in text-based image editing.

RL in Image Generation. Recent works have demonstrated reinforcement learning (RL) as a powerful paradigm for enhancing text-to-image generation [41, 23, 7, 29]. Pioneering works [24, 10, 46, 29] establish RL frameworks that optimize text prompts through iterative reward feedback, effectively aligning the generated images with the aesthetic score and semantic metric. By designing domain-specific reward mechanisms, these methods could effectively generate high quality images aligning with reward function. Inspired by this line of research but differently, we propose formulating an RL environment for the image editing problem to identify the optimal hyperparameters.

3 Method

3.1 Preliminaries

Diffusion Models. Diffusion Models [13, 28, 39, 34] generate images by iteratively denoising Gaussian noise. They consist of:

`Forward Process:` Add noise to a clean image x_0 over T steps:

$$x_t \sim \mathcal{N}\big(\sqrt{\overline{\alpha}_t}\, x_0,\ (1-\overline{\alpha}_t)\mathbf{I}\big),$$

with $\overline{\alpha}_t$ decreasing from 1 to 0, thus $x_T \sim \mathcal{N}(0, \mathbf{I})$.

`Reverse Process:` Learn backward transitions

$$p(x_{t-1}\,|\,x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t),\ \sigma_t^2\mathbf{I}\big),$$

where

$$\mu_\theta(x_t, t) = \tfrac{1}{\sqrt{\alpha_t}}\Big(x_t - \epsilon_\theta(x_t, t)\,\tfrac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\Big), \quad \alpha_t = \tfrac{\overline{\alpha}_t}{\overline{\alpha}_{t-1}}.$$

The noise prediction network ϵ_θ is trained by

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{x_0,\,\epsilon\sim\mathcal{N}(0,\mathbf{I}),\,t}\big\|\epsilon - \epsilon_\theta(x_t, t)\big\|_2^2.$$

Image Editing. Image editing [5, 12, 45, 17, 40, 1] modifies images to align with editing prompts while preserving background. Under the prompt-to-prompt edit setting, we denote the source image I^{src}, source prompt p_{src}, edit prompt p_{edit}, and edit region mask M.

The editing pipeline contains two stages: 1. **Inversion:** From source image I^{src} and source prompt p_{src}, we extract noise set using an inversion technique. 2. **Denoising:** Using the editing prompt p_{edit} and extracted noise set, we gradually denoise the edited image I^{edit}. During the above denoising stage, the users often need to **search for optimal hyperparameter** (e.g. attention weights, inversion steps) to obtain the edited image aligning to p_{edit} and preserving background. Formally, for editing method $\mathcal{E}$ with hyperparameter set $\mathcal{H}$, we need to find

$$\mathcal{H}^* = \arg\min_{\mathcal{H}}\ \mathcal{D}\big(I^{edit}(\mathcal{E}(I^{src}, p_{edit}, \mathcal{H})),\ I^{target}\big),$$

where $\mathcal{D}$ measures how well I^{edit} matches the edit criteria.

Proximal Policy Optimization (PPO). PPO [37] is a popular and reliable method in Reinforcement Learning, and is widely applied in LLM training [30], prompt optimization [24, 10]. PPO is a policy gradient method, where the policy model is updated by the expected gradient of the policy output and the advantage. The common technique in Policy Gradient is to train two seperate networks for

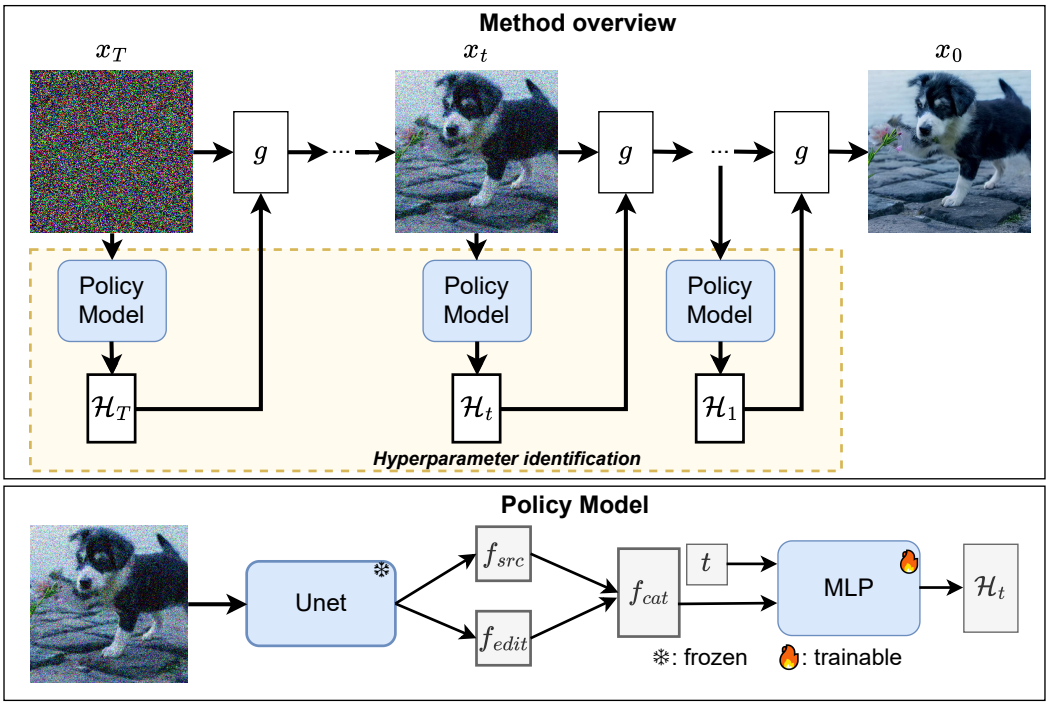


Figure 2: **Top**: Overview of the proposed AutoEdit framework. A policy model is injected to predict the step-wise hyperparameter $\mathcal{H}_t$ at each denoising step t. The predicted $\mathcal{H}_t$ is used with the one-step denoising function g and current state x_t to estimate x_{t-1}. **Bottom**: Architecture of the policy model. Features from the U-Net encoder under the original and edited prompts are extracted and concatenated, followed by several trainable layers to predict the policy output.

estimating the policy output π_θ (policy model) and the other network to estimate the reward to go V_τ (value model). In PPO, the policy model is updated by:

$$L(\theta) = E_t[\min(r_t A_t, \text{clip}(r_t, 1-\epsilon, 1+\epsilon)A_t)]$$

where A_t is the advantage computed at time t of the rollout process, and $r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the ratio between current and old policy. The value model is updated by:

$$L(\tau) = E_t[(V_\tau(s_t) - \hat{R}_t)^2]$$

where $\hat{R}_t$ is the expected return, which can be computed through the GAE method.

Hyperparameter Selection. Given specific p_{edit} and I^{src}, editing performance heavily depend on hyperparameters $\mathcal{H}$. Naive trial-and-error tuning runs denoising for each $\mathcal{H}$, which is infeasible as k grows. Previous methods like OIR [45] perform T editing trials for $r \in \{1, 2, ..., T\}$, but (1) searching cost scales to $\mathcal{O}(T^2)$ NFEs and (2) they tune the only hyperparameter: the inversion timestep. We introduce AutoEdit, a single-pass denoising framework that efficiently finds $\mathcal{H}^*$.

3.2 AutoEdit: Automated Hyperparameter Identification

3.2.1 RL Environment Formulation

We formulate the problem of optimizing hyperparameters as a RL task by integrating the RL framework into the diffusion denoising process. In this section, we define the key components of the RL environment:

State. In RL environment, we define $s_t = (x_t, t)$ as the state, with x_t is noisy latent at time $t = T \to 1$. The initial state is $s_T = (x_T, T)$

Action. At each state s_t at timestep t, selecting a hyperparameter configuration is treated as an action. Given a set of K hyperparameters $\mathcal{H} = \{h^1, h^2, \ldots, h^K\}$, we aim to determine the optimal configuration $\mathcal{H}_t = \{h_t^k\}_{k=1}^K$ at every s_t to maximize editing performance. This naturally leads us

to model each hyperparameter h^k as a temporal sequence $\{h_t^k\}_{t=T\to 1}$ across the editing trajectory. As a concrete example, consider the case where $\mathcal{H} = \{r\}$, with r denoting the inversion timestep. The editing process can be divided into two consecutive stages: denoising from T to r using the source prompt p_{src}, followed by denoising from r to 1 using the edit prompt p_{edit}. In this formulation, choosing r is equivalent to defining a prompt sequence $\mathcal{H}_t = \{h_t\}$, where $h_t \in \{p_{\text{src}}, p_{\text{edit}}\}$ depending on whether $t > r$ or $t \leq r$. Other hyperparameters can be similarly represented as time-dependent sequences similar the inversion timestep r, as elaborated in the Appendix. Therefore, at each timestep t, the configuration $\mathcal{H}_t$ constitutes a set of parallel actions over all K hyperparameters at state s_t, enabling flexible and fine-grained control throughout the editing process. Once the hyperparameter configuration $\mathcal{H}_t$ is defined, the next state x_{t-1} can be found by applying the diffusion denoising step $x_{t-1} = g(x_t, t, \mathcal{H}_t)$, where g is one-step denoising function.

Reward. We define the reward function to align with the measure $\mathcal{D}$, which is computed from the edited image I^{edit} based on two editing criteria: background preservation and prompt alignment.

Background preservation. The background region, defined as the area outside the editing mask M, should remain consistent with the original image I^{src}. We employ the mean squared error (MSE) to quantify this consistency since it is susceptible to subtle pixel-level changes. The background preservation reward is formulated as:

$$R_{noedit} = -\text{MSE}((1 - \text{M}) \odot I^{src}, (1 - \text{M}) \odot I^{edit}), \tag{1}$$

where $\odot$ denotes element-wise multiplication.

Prompt Alignment. The content within the mask M of the edited image I^{edit} should be semantically aligned with the edit prompt p_{edit}. We measure this alignment using the CLIP score [33], which computes the semantic similarity between the CLIP embeddings of the edited image and the text prompt. Denote R_{edit} as the reward of the prompt alignment.

The total reward is computed as a weighted combination of the prompt alignment and background preservation rewards:

$$R(\mathcal{H}, I^{src}, p_{src}, p_{edit}) = \alpha R_{edit} + \beta R_{noedit}, \tag{2}$$

where α and β are coefficients that govern the trade-off between edit effectiveness and background preservation. Empirically, we set $\alpha = \beta = 30$ as the default configuration to achieve an optimal equilibrium between the two objectives. This balanced weighting ensures robust alignment with the edit prompt while minimizing undesired alterations to the background. The sensitivity of model behavior to variations in α and β, along with further analysis of this trade-off, is detailed in Section 4.4.

Global editing and small region editing For global editing tasks, such as style transfer, we set the mask to $M = 1$ across the entire image and define the reward as $R = R_{edit}$, following the evaluation protocol of the PieBench dataset. For small region editing, such as changing eye color, the CLIP score reward is less effective on small regions. To address this, we incorporate a Large Vision-Language Model (LVLM) to compute prompt alignment rewards. Specifically, for each edited image, given the original and edited prompts, we pose a question related to the editing instruction and let the model select the correct answer from multiple choices. The reward is defined as the total number of correct answers. We provide the ablation study and the effectiveness of using LVLM as the reward compared to the CLIP score reward in section 4.4.

Termination. The RL process terminates upon completing all T denoising steps. We employ Proximal Policy Optimization (PPO) [37] for training, which comprises two core components: 1) a policy model $\pi_\theta(s_t) = \pi_\theta(x_t, t)$ that estimates action probabilities given states s_t, and 2) a value model $V_\tau(x_t, t)$ that predicts state rewards. Additional implementation details are provided in Section 3.2.3.

We frame hyperparameter optimization as a reinforcement learning problem where each denoising timestep corresponds to a unique state. This formulation induces an exponential state space growth (N^T possible states), where N represents the possible value of state. To address the challenge of this huge state space while leveraging the constrained range of the hyperparameters in image editing, we propose a two-phase RL strategy: **Phase 1 - Pretraining:** The policy is pretrained using hyperparameters randomly sampled from a predefined prior distribution. This constrains exploration to a promising subspace, significantly reducing state space complexity. The resulting model π_{θ_1} serves as initialization for subsequent learning. **Phase 2 - Online Learning:** Starting from π_{θ_1}, the policy π_{θ_2} engages in environment interactions while continuously optimizing through

reward maximization. This stage enables optimal hyperparameter searching within the constrained hyperparameter space from **Phase 1**.

3.2.2 Phase 1 - Policy Initialization

In image editing, we exploit known priors for each hyperparameter, *e.g.*, inversion timesteps [15] commonly in $[20, 45]$ or attention-replacement ratios [12] commonly in $[0.2, 0.8]$. Let $p_0(\mathcal{H})$ denote these priors over $\mathcal{H} = \{\mathcal{H}_T, \ldots, \mathcal{H}_1\}$. We train the policy model π_θ to align its outputs with p_0 by minimizing the objective function:

$$\theta_1 = \arg\min_{\theta} \; \mathbb{E}_{\mathcal{H}\sim p_0(\mathcal{H})}\left[\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_1\big(\pi_\theta(x_t, t),\, \mathcal{H}_t\big)\right], \tag{3}$$

where $\mathcal{L}_1$ penalizes deviation from the sampled $\mathcal{H}_t$.

3.2.3 Phase 2 - Hyperparameters Identification

During online exploration, at each denoising step t, the policy samples $\mathcal{H}_t \sim \pi_\theta(x_t, t)$ (see Figure 2). This action reconstructs the previous latent $x_{t-1} = g(x_t, t, \mathcal{H}_t)$, advancing the denoising process. The full hyperparameter sequence $\mathcal{H} = \{\mathcal{H}_T, \ldots, \mathcal{H}_1\}$ yields the edited image I^{edit}, from which we compute a reward R in equation 2. We then update the policy by minimizing the following objective:

$$\mathcal{L}_2 = -\mathbb{E}_{\substack{\mathcal{H}_t\sim\pi_\theta(x_t,t) \\ (I^{src},p_{src},p_{edit})\sim p_{data}}}\big[R(\mathcal{H}, I^{src}, p_{src}, p_{edit}) - \eta\, D_{KL}\big], \tag{4}$$

where p_{data} is the data distribution, and D_{KL} regularizes the current policy A_{θ_2} against the stage-1 policy π_{θ_1} [30, 24, 20]. Since $\mathcal{H}_t \sim \pi_\theta(x_t, t)$, we only require $O(1)$ for each denoising step, resulting in $\mathcal{O}(T)$.

We also train the value model V_τ to predict the cumulative reward at each state by minimizing the error between its estimation and the actual return. In PPO fashion, the policy π_θ and value model V_τ are alternately optimized to maximize expected cumulative reward. (More information of training and testing phases could be found in the Appendix)

3.2.4 Network Design

We treat each pair (x_t, t) as an RL state. We condition the policy on x_t, timestep t, and prompts p_{src}, p_{edit} since the optimial hyperparameters depend on the provided prompts, which match with the U-Net input. A pretrained U-Net Encoder U_ω extracts two features $f_{src} = U_\omega(x_t, t, p_{src}), \quad f_{edit} = U_\omega(x_t, t, p_{edit})$. These feature maps are concatenated, then passed through a 2×2 convolution and spatial average pooling layer to yield f_{cat}. Timestep t is encoded sinusoidally and linearly projected to f_t. We concatenate $[f_{\text{cat}}; f_t]$, process it with two ReLU-activated fully connected layers, then use K separate linear heads to predict K hyperparameters. The value model shares this backbone architecture but output a single scalar output. Only the newly added layers are trained (see Fig. 2).

4 Experiment

4.1 Dataset Setup

Our training data originates from the EditBench collection [22], with task selection strictly aligned to PieBench [16] benchmark objectives. To establish a coherent framework for prompt-to-prompt editing, each training datapoint requires four elements: the original image, the original prompt (generated by ChatGPT based on image content), the edit prompt (derived through ChatGPT analysis of edited images and instruction prompts), and the edit mask (object segmentation via SAM [19]). This curation process involves three systematic steps: **Object Localization**: ChatGPT extracts target editing objects from instruction prompts through cross-modal analysis. **Prompt Engineering**: Dual prompts (original/edited) are synthesized by ChatGPT using image-object contextual relationships. **Mask Generation**: SAM precisely segments identified objects from original images. The resulting dataset contains 2,000 rigorously annotated samples. For evaluation, we adopt the PieBench dataset [16], which contains 700 samples across diverse editing scenarios, ensuring comprehensive capability assessment.

4.2 Implementation Details

We choose denoising step $T = 50$ by default. During **Phase-1** training, we implement parameter initialization policies based on different editing method requirements. 1) For methods requiring inversion timestep searching, we randomly sample the inversion timestep within the range of $0.35 - 0.95$ of the total denoising timesteps T; 2) For methods involving cross-attention or self-attention replacement ratios, we initialize these parameters by uniform sampling from their default range of $0.2 - 0.8$; 3) For scalar hyperparameters (*e.g.*, attention weights in [12]), we instruct the policy model to predict their default values. More details of the prior for each type of hyperparameter can be found in the Appendix. In **Phase-2**, we integrate the **Phase-1** model into the training framework to compute the KL divergence term D_{KL} in Equation 4, following the implementation strategies in [24, 30]. Both policy and value models are optimized using Adam [18] with a fixed learning rate of 5×10^{-5}. For the reward function configuration, we set the coefficients $\alpha = 30$ and $\beta = 30$ as default values unless otherwise specified. Consistent with prior works [24, 30], the KL divergence coefficient remains $\gamma = 0.02$ across all experiments.

4.3 AutoEdit Improves the Performance over Other Methods

Baselines. We evaluate AutoEdit across multiple image editing frameworks, including training-free methods such as DDIM-Inversion [39], DDPM-Inversion [15], PnP-Inversion [16], Prompt-to-Prompt [12], MasaCtrl [5], and Null-text Inversion [25], as well as training-based methods such as InstructPix2Pix [4] and UltraEdit [47]. Our experiments span a variety of base models, including SD 1.4, SD 1.5, SDXL, and DiT. The hyperparameter settings for each method such as inversion timestep ranges and attention replacement ratios are systematically reported in the Appendix.

Metrics. Following the PieBench benchmark [16], our evaluation protocol measures four key aspects of editing performance: (i) structural consistency, quantified by Structure Distance (SD); (ii) background preservation, assessed with classical metrics including PSNR, SSIM, MSE, and LPIPS; and (iii) semantic alignment, evaluated via CLIP scores (ViT-B/32) computed separately for edited regions and full images. To further validate the quality of edits, particularly on small regions, we additionally incorporate an evaluation based on the judgment of a Large Vision-Language Model (LVLM), referred to as the LLM Score. For each image and its corresponding editing instruction, the LVLM is prompted with a question derived from the instruction, and its response is used to quantify editing quality. Details of question construction and the LLM Score evaluation protocol are provided in the Appendix.

Quantitative Results. We present the main experimental results in Table 1. We report the best performance for the baseline methods with the hyperparameter configuration that yields the highest reward. For DDIM Inversion [39] and DDPM Inversion [15], the best configuration of the inversion timestep r is 35 and 40, respectively. Under these configurations, the models achieve high CLIP scores but struggle with maintaining background consistency. When integrated with AutoEdit, both methods demonstrate significantly improved background preservation, accompanied by only a slight drop in CLIP scores. We contend that this minor decrease has minimal impact on editing fidelity due to the inherent insensitivity of the CLIP metric. Moreover, these methods inherently face a trade-off between editing fidelity and background preservation. Our proposed reward addresses this issue by offering a more balanced assessment of editing performance. Additional visual evidence of this trade-off is provided in the Appendix. Furthermore, AutoEdit consistently improves performance on the LLM Score metric for both inversion-based methods [39, 15]. For P2P [12], PnP [16], and MasaCtrl [5], AutoEdit leads to enhanced performance in both background preservation and CLIP editing scores, underscoring its effectiveness in guiding hyperparameter identification.

We further evaluate training-based approaches, including InstructPix2Pix [4] and UltraEdit [47], by applying AutoEdit to adaptively determine the CFG coefficient during the denoising process. Compared to the default fixed value of 7.5, the adaptive CFG strategy improves background preservation metrics while maintaining prompt alignment performance.

Our experiments span a diverse set of editing methods and base models, demonstrating the generalization of AutoEdit. More results of AutoEdit on flow-based editing method is in Table 6.

Qualitative Results. We present qualitative results in Figure 3. For each baseline, we visualize the output generated using its default hyperparameters, as specified in Table 1. Overall, AutoEdit consistently improves the performance of all editing methods by selecting more suitable hyperparameter

Table 1: **The comparison with popular image editing methods**. Note that AutoEdit enables improvement over baselines with only negligible cost.

Method	Base Model	Structure Distance ↓	Background Preservation PSNR ↑	SSIM ↑	MSE ↓	LPIPS ↓	CLIP Score Edited ↑	Whole ↑	LLM Score
DDIM-Inversion [39]	SD 1.4	38.10	21.36	76.67	103.95	146.60	**23.30**	**26.31**	0.96
+ AutoEdit		**18.74**	**24.65**	**81.28**	**52.94**	**95.10**	22.65	25.72	**1.12**
DDPM-Inversion [15]	SD 1.4	22.12	22.66	78.95	53.33	67.66	**23.02**	**26.22**	1.03
+ AutoEdit		**12.65**	**27.25**	**85.17**	**31.18**	**50.51**	22.52	25.83	**1.17**
PnP Inversion [16]	SD 1.5	11.65	27.22	84.76	35.86	60.67	22.10	25.02	1.10
+ AutoEdit		**11.06**	**27.85**	**85.04**	**33.77**	**60.12**	**23.00**	**25.79**	**1.19**
P2P [12]	SD 1.4	14.75	25.82	84.02	40.93	61.78	22.29	25.44	1.08
+ AutoEdit		**13.76**	**26.45**	**84.08**	**36.24**	**60.60**	**23.88**	**26.55**	**1.22**
MasaCtrl [5]	SD 1.4	28.38	22.17	79.67	86.97	79.67	21.16	23.96	0.92
+ AutoEdit		**21.33**	**23.48**	**80.06**	**46.28**	**71.35**	**21.75**	**24.86**	**0.99**
DDPM-Inversion [15]	SDXL	7.12	26.13	89.88	35.32	65.62	**23.0**	**27.11**	1.19
+AutoEdit		**6.46**	**27.86**	**90.50**	**20.44**	**53.51**	22.9	26.7	**1.27**
UltraEdit [47]	MM-DiT	10.82	26.5	84.7	46.7	75.8	22.4	25.6	1.20
+AutoEdit		**7.61**	**27.3**	**86.2**	**37.6**	**64.9**	**22.6**	**25.7**	**1.26**
InstructPix2Pix [4]	SD 1.5	35.37	20.8	76.4	226.8	157.3	22.1	24.5	0.65
+AutoEdit		**28.68**	**22.2**	**78.5**	**181.4**	**132.8**	**22.3**	**24.7**	**0.82**
Null-text [25]	SD 1.4	19.87	23.8	79.9	64.4	109.8	22.3	25.9	1.12
+AutoEdit		**10.91**	**25.7**	**82.4**	**45.4**	**82.3**	**22.6**	**26.3**	**1.21**

configurations. Our qualitative analysis highlights several common failure cases resulting from suboptimal hyperparameter choices: inaccurate background reconstruction (*e.g.*, the stone background in the cat image for DDIM-Inversion and DDPM-Inversion), unnatural object synthesis (*e.g.*, the distorted rock in P2P), failure to execute the desired edit (*e.g.*, the unaltered stone in MasaCtrl or the unchanged noodles in MasaCtrl, PnP, and DDIM-Inversion), and noticeable facial discrepancies (*e.g.*, the altered woman's face in DDPM-Inversion and P2P). These findings demonstrate that improved hyperparameter selection, facilitated by AutoEdit, can substantially enhance editing quality, even for weak baseline methods such as DDIM-Inversion.

Policy Model Behavior in Inference. We analyze the behavior of key hyperparameters predicted by the policy model during inference. For the inversion timestep t in DDPM-Inversion [15], where the policy chooses between p_{src} and p_{edit}, we define the inversion timestep as the first step t where the policy selects p_{edit}. Figure 4a shows the distribution of inversion timesteps chosen by the policy model. We observe that the policy most frequently selects t between 25 and 40, which aligns with common choices in prior editing methods. A similar analysis is conducted for the cross-attention replacement ratio r in P2P [12], with results shown in Figure 4b. We find that r is most often selected between 0.3 and 0.6. Importantly, both t and r vary significantly across samples, indicating that AutoEdit adapts its hyperparameter predictions for each specific image to optimize the final reward.

Reward Across Training Episodes. We plot the reward of AutoEdit on DDPM-Inversion [15] over the first 3000 training episodes. The results are shown in Figure 4c, where episode 0 corresponds to the **Phase-1** model A_{θ_1}. We observed a consistent increase in reward during PPO training up to episode 2500, after which it slightly declined at episode 3000. Based on this trend, we limit training to the first 2500 episodes (approximately 15 epochs).

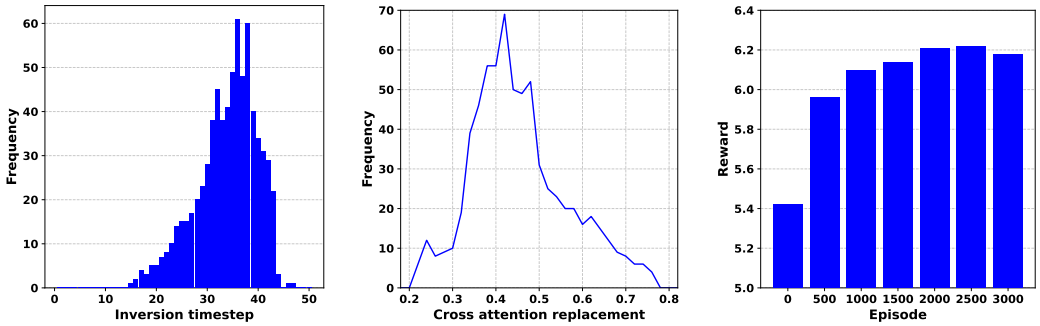
4.4 Ablation Study

Table 2: The necessity of **Phase-1** (policy initialization).

P1	P2	PSNR ↑	SSIM ↑	MSE ↓	LPIPS ↓	Edited ↑	Whole ↑	Reward
	✓	18.2	74.5	208.7	57.9	**23.2**	**26.3**	6.12
✓		22.1	77.4	52.7	69.7	20.7	23.4	5.42
✓	✓	**27.2**	**85.3**	**31.1**	**50.5**	22.5	25.8	**6.25**

Effect of Reward Coefficients of α and β. We investigate the impact of different values of α and β in the reward function. Using DDPM-Inversion [15] as the baseline, we fix $\alpha = 30$ and train AutoEdit with varying values of β. All models are initialized from the same **Phase-1** checkpoint A_1. Table 3 summarizes the behavior of AutoEdit under different β settings. When β is small (*e.g.*, $\beta = 10$ or $\beta = 20$), the reward function

Figure 3: We compare the qualitative results of AutoEdit with the default hyperparameter choice of the baseline. Overall, AutoEdit can search for better hyperparameters, resulting in better object editing, background preservation, and more natural images.



(a) (b) (c)

Figure 4: The analysis of a) inversion timestep, b) cross-attention replacement, and c) the reward during the training of the policy model A_θ.

emphasizes the CLIP score more. As a result, AutoEdit achieves higher CLIP scores, but at the cost of reduced background preservation. Conversely, when β is increased to 40, the model prioritizes background preservation, as it becomes more influential in the reward signal. $\beta = 30$ strikes a good balance between CLIP alignment and background consistency. Figure 5 provides qualitative examples illustrating the effects of different β values.

Different choice of the reward. We find that the CLIP score performs poorly on small region image editing tasks with a limited edit mask M, such as eye color modification. As a result, using CLIP as the reward fails to reliably capture editing quality. In contrast, Large Vision-Language Models (LVLMs) can effectively assess localized edits, for example, verify whether the eye color has been correctly changed, as highlighted by recent editing benchmarks [22, 44]. Since our proposed RL framework is generic, it can incorporate any reward function that reflects editing quality. Accordingly, we replace the CLIP-based reward with an LVLM-based evaluation. Our LVLM reward is similar to

Figure 5: Edited image with different value of β

Table 3: Comparison of the behavior of AutoEdit with the different choices of α and β. When we decrease the β, AutoEdit optimizes the CLIP score. Otherwise, it tries to preserve the background.

α, β	**PSNR** ↑	**SSIM** ↑	**MSE** ↓	**LPIPS** ↓	**Edited** ↑	**Whole** ↑
$\alpha = 30, \beta = 10$	19.65	77.11	150.5	138.6	24.15	27.34
$\alpha = 30, \beta = 20$	23.59	82.15	66.84	82.30	23.44	26.95
$\alpha = 30, \beta = 30$	27.25	85.17	31.18	50.51	22.52	25.83
$\alpha = 30, \beta = 40$	28.53	86.03	24.72	42.80	21.36	24.36

the LLM Score metric, which is described in detail in the Appendix. Table 4 reports the performance of AutoEdit under LVLM evaluation. We observe consistent improvements in both background preservation metrics and LLM Score, while maintaining strong prompt alignment performance.

Policy Initialization. We conduct experiments to highlight the importance of **Phase-1** training (policy initialization). Using DDPM-Inversion [15] as the baseline, we compare results with and without **Phase-1**. The outcomes are summarized in Table 2. As shown in the table, omitting **Phase-1** causes the model to apply overly strong edits while failing to preserve the background, resulting in a substantially lower background preservation score. We attribute this behavior to the difficulty of exploring the large state space introduced by the 50-step denoising process in DDPM. In contrast, incorporating **Phase-1** improves background preservation and CLIP alignment, demonstrating the effectiveness of this initialization phase in guiding policy learning.

Table 4: Comparison of AutoEdit performance with LLM Score as the reward function

Method	**PSNR**	**SSIM**	**MSE**	**LPIPS**	**Edited**	**Whole**	**LLM**
DDPM Inv	26.1	89.8	35.3	65.6	23.0	27.1	1.19
+ AutoEdit	27.8	90.5	20.4	53.5	22.9	26.7	1.27
+ AutoEdit + LLM	**29.1**	**91.8**	**19.1**	**49.1**	22.7	26.6	**1.31**

Computational Cost. We compare the inference time and number of parameters between the original U-Net and AutoEdit for a single image-prompt pair in DDPM-Inversion. Table 5 reports the inference time of AutoEdit on Stable Diffusion. Our method introduces only negligible overhead in inference time and model size compared to the original U-Net.

Table 5: Computational cost of AutoEdit is negligible.

Method	Inference (ms)	#Params
SD Unet	20.89	86M
+AutoEdit	21.03 (+0.6%)	87.1M (+0.6%)

5 Conclusion

We propose AutoEdit, an RL-based framework for per-image hyperparameter identification in diffusion-model editing. By casting the denoising process as a reinforcement learning problem, AutoEdit searches within a single trajectory, eliminating the exponential overhead of trial-and-error. Empirical results across multiple editing methods demonstrate that AutoEdit improves editing quality while maintaining runtime efficiency.

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] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[3] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

[4] 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.

[5] Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22560–22570, 2023.

[6] Yingying Deng, Xiangyu He, Changwang Mei, Peisong Wang, and Fan Tang. Fireflow: Fast inversion of rectified flow for image semantic editing. *arXiv preprint arXiv:2412.07517*, 2024.

[7] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023.

[8] Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Yang Wang, Yinfei Yang, and Zhe Gan. Guiding instruction-based image editing via multimodal large language models. *arXiv preprint arXiv:2309.17102*, 2023.

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

[10] Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:66923–66939, 2023.

[11] Xiaoxiao He, Ligong Han, Quan Dao, Song Wen, Minhao Bai, Di Liu, Han Zhang, Martin Renqiang Min, Felix Juefei-Xu, Chaowei Tan, et al. Dice: Discrete inversion enabling controllable editing for multinomial diffusion and masked generative models. *arXiv preprint arXiv:2410.08207*, 2024.

[12] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross-attention control. In *ICLR*, 2023.

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

[14] Yuzhou Huang, Liangbin Xie, Xintao Wang, Ziyang Yuan, Xiaodong Cun, Yixiao Ge, Jiantao Zhou, Chao Dong, Rui Huang, Ruimao Zhang, et al. Smartedit: Exploring complex instruction-based image editing with multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8362–8371, 2024.

[15] 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.

[16] Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Pnp inversion: Boosting diffusion-based editing with 3 lines of code. In *The Twelfth International Conference on Learning Representations*.

[17] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6007–6017, 2023.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[20] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[21] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip HS Torr. Manigan: Text-guided image manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7880–7889, 2020.

[22] Yiwei Ma, Jiayi Ji, Ke Ye, Weihuang Lin, Zhibin Wang, Yonghan Zheng, Qiang Zhou, Xiaoshuai Sun, and Rongrong Ji. I2ebench: A comprehensive benchmark for instruction-based image editing. *arXiv preprint arXiv:2408.14180*, 2024.

[23] Zichen Miao, Jiang Wang, Ze Wang, Zhengyuan Yang, Lijuan Wang, Qiang Qiu, and Zicheng Liu. Training diffusion models towards diverse image generation with reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10844–10853, 2024.

[24] Wenyi Mo, Tianyu Zhang, Yalong Bai, Bing Su, Ji-Rong Wen, and Qing Yang. Dynamic prompt optimizing for text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26627–26636, 2024.

[25] 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.

[26] Chong Mou, Xintao Wang, Jiechong Song, Ying Shan, and Jian Zhang. Dragondiffusion: Enabling drag-style manipulation on diffusion models. *arXiv preprint arXiv:2307.02421*, 2023.

[27] Seonghyeon Nam, Yunji Kim, and Seon Joo Kim. Text-adaptive generative adversarial networks: manipulating images with natural language. *Advances in neural information processing systems*, 31, 2018.

[28] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[29] Owen Oertell, Jonathan Daniel Chang, Yiyi Zhang, Kianté Brantley, and Wen Sun. Rl for consistency models: Reward guided text-to-image generation with fast inference. In *Reinforcement Learning Conference*, 2024.

[30] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[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] Hao Phung, Quan Dao, and Anh Tran. Wavelet diffusion models are fast and scalable image generators. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10199–10208, 2023.

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[34] 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.

[35] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510, 2023.

[36] 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.

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[38] Yujun Shi, Chuhui Xue, Jun Hao Liew, Jiachun Pan, Hanshu Yan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8839–8849, 2024.

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

[40] Linoy Tsaban and Apolinário Passos. Ledits: Real image editing with ddpm inversion and semantic guidance. *arXiv preprint arXiv:2307.00522*, 2023.

[41] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8228–8238, 2024.

[42] Jiangshan Wang, Junfu Pu, Zhongang Qi, Jiayi Guo, Yue Ma, Nisha Huang, Yuxin Chen, Xiu Li, and Ying Shan. Taming rectified flow for inversion and editing. *arXiv preprint arXiv:2411.04746*, 2024.

[43] Weihao Xia, Yujiu Yang, Jing-Hao Xue, and Baoyuan Wu. Tedigan: Text-guided diverse face image generation and manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2256–2265, 2021.

[44] Zitong Xu, Huiyu Duan, Bingnan Liu, Guangji Ma, Jiarui Wang, Liu Yang, Shiqi Gao, Xiaoyu Wang, Jia Wang, Xiongkuo Min, et al. Lmm4edit: Benchmarking and evaluating multimodal image editing with lmms. *arXiv preprint arXiv:2507.16193*, 2025.

[45] Zhen Yang, Ganggui Ding, Wen Wang, Hao Chen, Bohan Zhuang, and Chunhua Shen. Object-aware inversion and reassembly for image editing. In *The Twelfth International Conference on Learning Representations*.

[46] Yinan Zhang, Eric Tzeng, Yilun Du, and Dmitry Kislyuk. Large-scale reinforcement learning for diffusion models. In *European Conference on Computer Vision*, pages 1–17. Springer, 2024.

[47] Haozhe Zhao, Xiaojian Shawn Ma, Liang Chen, Shuzheng Si, Rujie Wu, Kaikai An, Peiyu Yu, Minjia Zhang, Qing Li, and Baobao Chang. Ultraedit: Instruction-based fine-grained image editing at scale. *Advances in Neural Information Processing Systems*, 37:3058–3093, 2024.

A AutoEdit baselines

In this section, we provide the details of the hyperparameters used in each baseline method:

Baselines

DDIM/DDPM Inversion [39, 15]: These methods first invert the image to a latent representation x_r using the source prompt p_{src}, followed by a denoising process from timestep r to 0 conditioned on the edit prompt p_{edit}. The only required hyperparameter in this process is the inversion timestep, i.e., $\mathcal{H} = \{r\}$. Due to the reversibility of the DDIM/DDPM framework, each denoising step can be viewed as an action $\mathcal{H}_t = \{h_t\}$, where $h_t \in \{p_{src}, p_{edit}\}$. By default, the inversion timestep is set to $r = 35$ for DDIM-Inversion and $r = 40$ for DDPM-Inversion.

P2P [12]: introduces three key hyperparameters: the inversion timestep r, the cross-attention replacement ratio u, and the attention weight w, summarized as $\mathcal{H} = \{r, u, w\}$. The inversion timestep r can be handled similarly to the strategy used in DDIM/DDPM-Inversion. The cross-attention ratio u modifies the attention mechanism in the U-Net by replacing the cross-attention conditioned on p_{edit} with that of p_{src} during the first $u \cdot T$ denoising steps, where T is the total number of steps. This decision is modeled as a binary action at each timestep: whether to replace the cross-attention or not. The attention weight w scales the cross-attention values for the edited word, determining the emphasis placed on that word in the output image. We discretize w by allowing the policy to select from a fixed set of values $\{0.5, 1.0, 1.5, 2.0, 3.0, 5.0\}$. Formally, we define the action space at each timestep as $\mathcal{H}_t = \{h_t^1, h_t^2, h_t^3\}$, where $h_t^1 \in \{p_{src}, p_{edit}\}$, $h_t^2 \in \{0, 1\}$ (with 1 indicating replacement), and $h_t^3 \in \{0.5, 1.0, 1.5, 2.0, 3.0, 5.0\}$. By default, the inversion timestep is set to $r = 10$, the cross-attention ratio to $u = 0.4$, and the attention weight to $w = 1.0$.

PnP Inversion [16]: In this method, two binary hyperparameters are defined at each denoising step, denoted as $\mathcal{H}_t = \{h_t^1, h_t^2\}$. The first, $h_t^1 \in \{0, 1\}$, determines whether to replace the self-attention computation in the edit branch with that from the unconditional branch. The second, $h_t^2 \in \{0, 1\}$, controls whether to replace the convolutional features of the edit branch with those from the unconditional branch. In the default setting of PnP Inversion, self-attention is replaced during the first 80% of the denoising process, while convolutional features are replaced during the first 50% of the steps.

MasaCtrl [5]: At each denoising step, the method decides whether to replace the self-attention in the edit branch with that from the unconditional branch. This decision is represented by a binary hyperparameter $\mathcal{H}_t = \{h_t\}$, where $h_t \in \{0, 1\}$. By default, the replacement is applied starting from timestep $t = 4$ of the denoising process.

UltraEdit [47] and ***InstructPix2Pix*** [4] Both methods are training-based editing method. Our RL framework is used to optimize the CFG coefficient during the sampling process of both methods. The default CFG coefficient is set to 7.5.

Null-text inversion [25] Null-text is an inversion method that addresses the mismatch between inversion and denoising process through CFG by optimizing the null embedding. In this method, we use P2P as the editing operation. We only use RL to optimize the inversion timestep and cross attention ratio u. Similar to P2P, we set the default value of $r = 40$ and $u = 0.4$.

Phase-1 prior for each type of hyperparameters: The inversion timestep r is randomly sampled from the range $[50 - 0.65T, 50 - 0.05T]$. The cross-attention ratio is similarly sampled from the range $[0.2, 0.6]$. The prior for the attention weight w is fixed at 1.0. For PnP Inversion [16], the ratios for replacing self-attention and convolutional features are randomly sampled from the range $[0.2T, 0.8T]$. Lastly, in MasaCtrl [5], self-attention replacement begins at timestep t, where t is randomly selected between step 4 and step 20.

Phase-1 loss: Since all the parameterizations of $\mathcal{H}_t$ are discrete, we choose the loss $\mathcal{L}_1$ for **Phase-1** training to be the cross-entropy loss. After sampling $\mathcal{H} \sim p_0(\mathcal{H})$, the loss $\mathcal{L}_1$ is applied at each timestep to enforce the policy model's prediction to match $\mathcal{H}_t$, the timestep-specific parameterization of $\mathcal{H}$ at t.

Table 6: Results of AutoEdit on a Flow-based image editing method

Method	PSNR	SSIM	CLIP Edit	CLIP Whole	LLM Score
Taming flow [42]	23.4	81.5	22.9	26.0	1.22
+AutoEdit	**25.7**	**85.2**	**23.4**	**26.1**	**1.30**
Fireflow [6]	23.1	82.2	22.4	25.2	1.20
+AutoEdit	**26.2**	**86.2**	**22.9**	**25.2**	**1.27**

B More results on flow-based editing

We present the results of AutoEdit applied to flow-based editing methods, as shown in Table 6. Specifically, AutoEdit is used to search for the optimal injection timestep, a key hyperparameter in both Taming Flow [42] and Fireflow [6]. Overall, AutoEdit enhances the baseline performance in terms of both background preservation and prompt alignment, demonstrating its effectiveness in automatically identifying hyperparameters that improve editing results.

C Global editing

For global editing tasks, such as style transfer, the editing mask is set to $M = 1$ for the entire image. In this case, $R_{noedit} = 0$, and the reward function R reduces to the CLIP score. To assess the effectiveness of AutoEdit in this setting, we report CLIP scores in Table 7, comparing against the baseline methods DDPM-Inversion [15] and P2P [12]. The results show that AutoEdit consistently improves editing performance on the style transfer task relative to the baselines.

Table 7: AutoEdit on global editing task

Base model	**Editing method**	**CLIP score**
SDXL	DDPM-Inv [15]	26.5
SDXL	+AutoEdit	**28.3**
SD 1.4	P2P	25.5
SD 1.4	+AutoEdit	**26.7**

D Convergence of PPO.

We evaluate how closely AutoEdit approaches the optimal reward through PPO training. In this experiment, we compare the reward achieved by AutoEdit against the best possible reward obtained from combinations of k different hyperparameter values, where $k \in \{1, 2, 3\}$. We compute the reward for each image using k different hyperparameter settings and select the maximum among them. The k values are selected from the set that yields the highest possible reward.

Table 8: Comparison of AutoEdit 's reward with baselines using multiple optimal hyperparameter sets. AutoEdit achieves performance comparable to the best of three hyperparameters, demonstrating its convergence to a near-optimal reward.

Method	#Trials			AutoEdit	Optimal
	1	2	3		
DDIM-Inversion	5.81	6.03	6.11	6.09	6.17
DDPM-Inversion	6.11	6.21	6.23	6.25	6.32
P2P	6.17	6.31	6.37	6.38	6.45
MasaCtrl	5.47	5.59	5.65	5.65	5.75

E LLM Score

For small-region editing, the CLIP score fails to reliably capture editing quality. As illustrated in Figure 6, when the objective is to change a dog's eye color to blue, computing the CLIP score on a small mask M with the prompt "blue eye" yields a lower score for the correctly edited image. This highlights the inaccuracy of CLIP-based rewards in such scenarios.

Previous evaluation benchmarks [22, 44] have adopted LLM judgment as an evaluation tool. A key advantage of LLM-based evaluation is its ability to verify whether small region edits are correct, as illustrated in Figure 6. Since our RL framework is generic, we replace the CLIP based reward with an LLM-based evaluation, which we denote as LLMScore. This score not only measures the quality

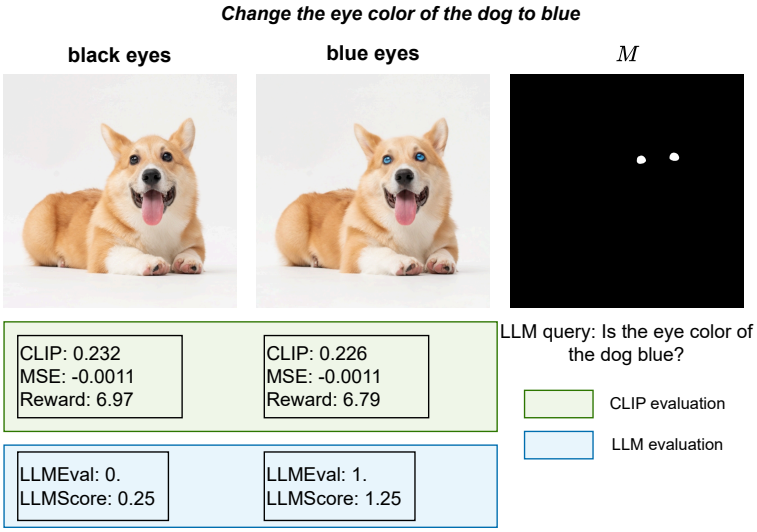


Figure 6: The advantage of LLM evaluation compared with CLIP score in small editing region

of an edited image but also serves directly as a reward function. More specifically, similar to the original reward, LLMScore consists of two components:

- **Prompt alignment** For each editing instruction, we construct a binary ("Yes/No") question to verify whether the required edit is present in the generated image. The edited image is then provided to the LLM, which outputs a response. We assign $R_{edit} = 1$ if the response is correct and $R_{edit} = 0$ otherwise.
- **Background preservation** For background preservation, we adopt MSE as the primary metric, as it is most sensitive to small changes in the background region. Accordingly, we define the background preservation reward as: $R_{noedit} = 1 - \gamma MSE((1-M) \odot I_{edit}, (1-M) \odot I_{src})$

The LLM score is computed as LLM Score $= R_{edit} + R_{noedit}$. We found $\alpha = 5$ works best in the editing task. For the LLM model, we adopt QwenVL-2.5-7B [2] as the LLM model.

F Trade-off between edit alignment and background preservation in DDIM/DDPM Inversion

In DDIM and DDPM Inversion [39, 15], a trade-off arises between alignment with the edit prompt and preservation of the original background. A high CLIP score typically indicates strong alignment with the edit prompt, but often at the cost of background fidelity, and vice versa. This trade-off is illustrated in Figure 7. Our proposed reward function offers a meaningful evaluation of edited images by explicitly balancing CLIP-based semantic alignment and background preservation (see Figure 7).

G Training and inference algorithm

We present the training and inference of AutoEdit in Algorithm 1, 2.

H Limitation

We illustrate the limitation of our approach in Figure 8. AutoEdit relies on the underlying editing method $\mathcal{E}$. Consequently, if $\mathcal{E}$ fails to perform a given edit, AutoEdit inherits this limitation. This issue is exemplified by the parrot case shown in Figure 8.

Figure 7: We provide the visualization of the trade-off between edit alignment and background preservation in DDIM-Inversion. Our reward can better reflect how good an edited image is.

Algorithm 1: AutoEdit Inference

Input: Editing method $\mathcal{E}$ with denoising function g and inversion function g^{-1}, input image I^{src}, origin and edit prompts p_{src}, p_{edit}, pretrained policy model π_θ, pretrained Stable Diffusion Model

Output: Edit image I^{edit}

for $t \leftarrow 1$ **to** T **do**

  Perform diffusion inversion: $x_{t+1}^{src} = g^{-1}(x_t, t, p_{src})$.

end

$x_T^{edit} = x_T^{src}$

for $t \leftarrow T$ **to** 1 **do**

  Sampling from the policy: $\mathcal{H}_t \sim \pi_\theta(x_t^{edit}, t)$.

  Denoise to get the previous noisy sample: $x_{t-1}^{edit} = g(x_t^{edit}, t, \mathcal{H}_t)$

end

VAE decode: $I^{edit} = VAE(x_0^{edit})$

return I^{edit};

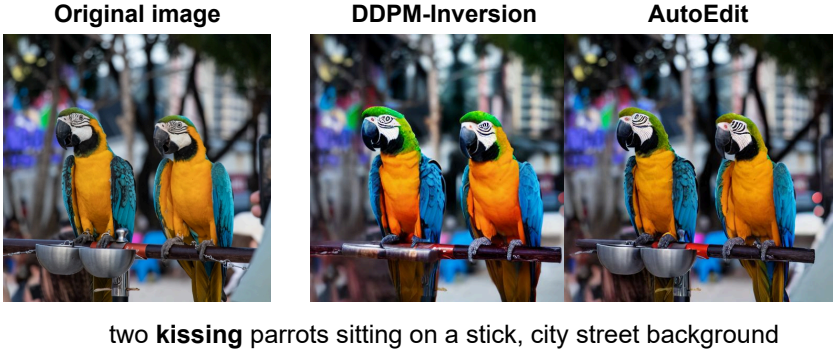


Figure 8: The limitation of our approach. A key limitation of our approach is that if the underlying editing method performs poorly on a given image, AutoEdit is also likely to inherit and suffer from the same image.

Algorithm 2: PPO Training of AutoEdit

Input: Editing method $\mathcal{E}$ with denoising function g and inversion function g^{-1}, policy model π_{θ_2}, **Phase 1** model π_{θ_1},pretrained Stable Diffusion Model, coefficient $\beta = 0.02, \gamma = 0.999, \lambda = 0.95, \epsilon = 0.2$

Inversion:

$I^{src}, p_{src}, p_{edit} \sim p_{data}$

for $t \leftarrow 1$ **to** T **do**

  Perform diffusion inversion: $x_{t+1}^{src} = g^{-1}(x_t, t, p_{src})$.

end

PPO exploration phase:

for $t \leftarrow T$ **to** 1 **do**

  Get the distribution of action at current policy model: $\pi_t^2 = \pi_{\theta_2}(x_t^{edit}, t)$.

  Get the distribution of action at **Phase-1** policy model: $\pi_t^1 = \pi_{\theta_1}(x_t^{edit}, t)$.

  Sampling the hyperparameter: $\mathcal{H}_t \sim \pi_t^2$.

  Compute the value: $v_t = V_\tau(x_t^{edit}, t)$.

  Denoise to get the previous noisy latent: $x_{t-1}^{edit} = g(x_t^{edit}, t, \mathcal{H}_t)$

end

Reward computation:

Get the edit image $I^{edit} = VAE(x_0^{edit})$

Compute the reward $R = \alpha R_{edit}(I^{edit}, p_{edit}, M) + \beta R_{noedit}(I^{edit}, I^{src}, M)$

Compute the return value and advantage recursively from the end of the denoising trajectory:

$A_0 = 0$

for $t \leftarrow 1$ **to** T **do**

  $\delta = R + \gamma v_{t-1} - v_t$

  Compute the advantage: $A_t = \delta + \gamma\lambda A_{t-1}$

  Compute the return: $r_t = A_t + v_t$

end

Training value model: $\mathcal{L}_{vf} = \mathbb{E}_t[(V_\tau(x_t, t) - r_t)^2]$

Training the policy model:

Compute ratio $u_t = \exp(\log \pi_{\theta_2}(x_t, t) - \log(\pi_t^2))$.

Clip objective: $\mathcal{L}_{clip} = \min(A_t u_t, A_t clip(u_t, 1-\epsilon, 1+\epsilon))$

Policy loss: $\mathcal{L}_{pg} = -\mathbb{E}_t[\mathcal{L}_{clip} - \beta D_{KL}(\pi_t^2 \| \pi_t^1)]$

Update θ_2, τ based on gradient optimization.

return π_{θ_2}

I User study

We conducted a user study involving 50 participants to evaluate the performance of AutoEdit. Each survey question presented a side-by-side comparison between AutoEdit and the same editing method with two different sets of hyperparameters. The survey comprised a total of 30 questions. Overall, 82.78% of the responses favored the results produced by AutoEdit over the baseline, indicating a clear human preference for the outputs generated by our method.

J Additional qualitative examples

We present additional qualitative examples comparing the editing baselines with and without AutoEdit in Figures 9, 10, 11, 12. Furthermore, we compare AutoEdit against the same methods using different hyperparameter settings in Figures 14 and 15. Overall, AutoEdit achieves consistently strong editing results across various hyperparameter configurations, thereby significantly reducing the manual tuning effort required by practitioners. We also include the qualitative results of AutoEdit applying to the Flux-based editing [42, 6] methods, as shown in Figure 13

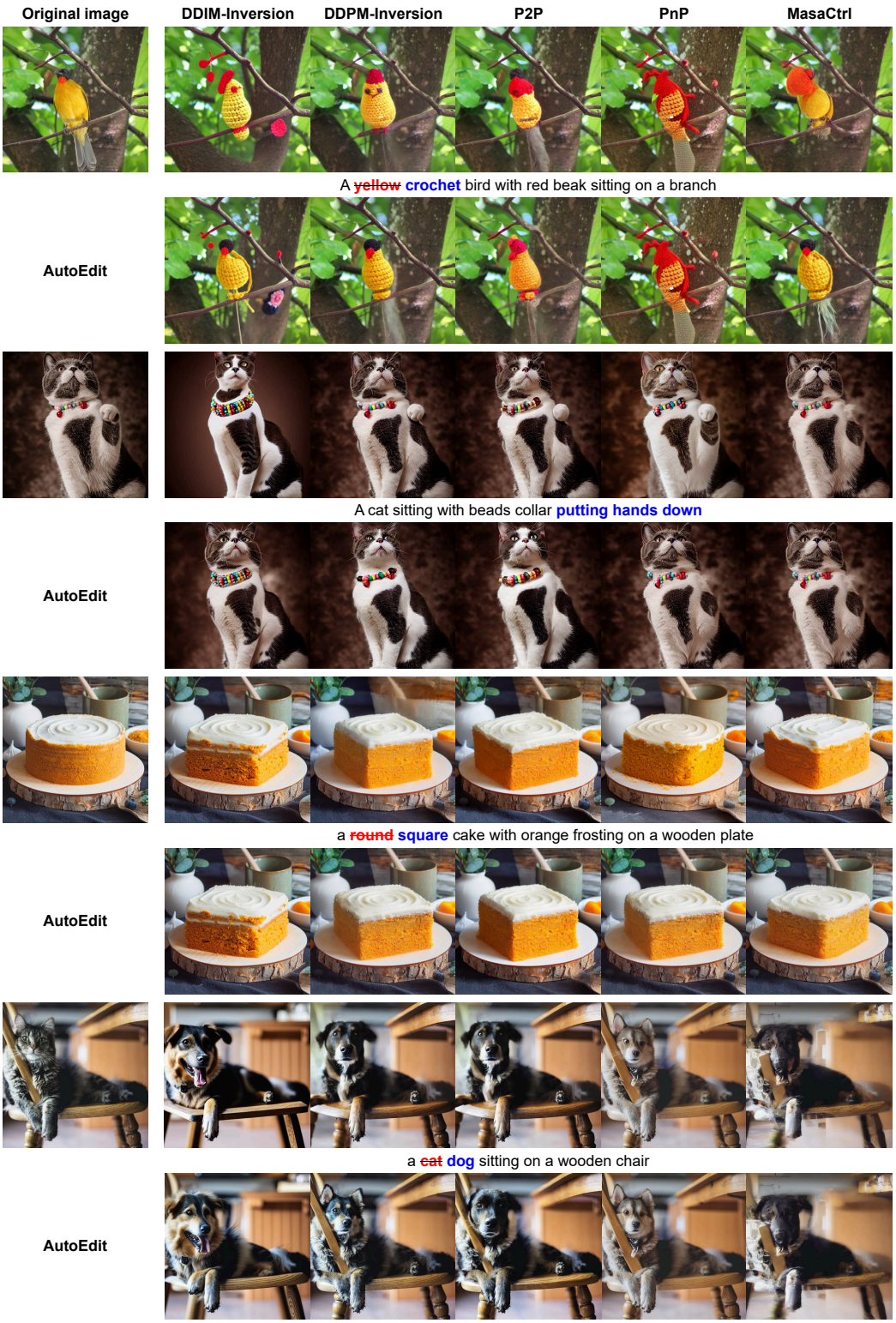


Figure 9: Additional qualitative samples

Figure 10: Additional qualitative results

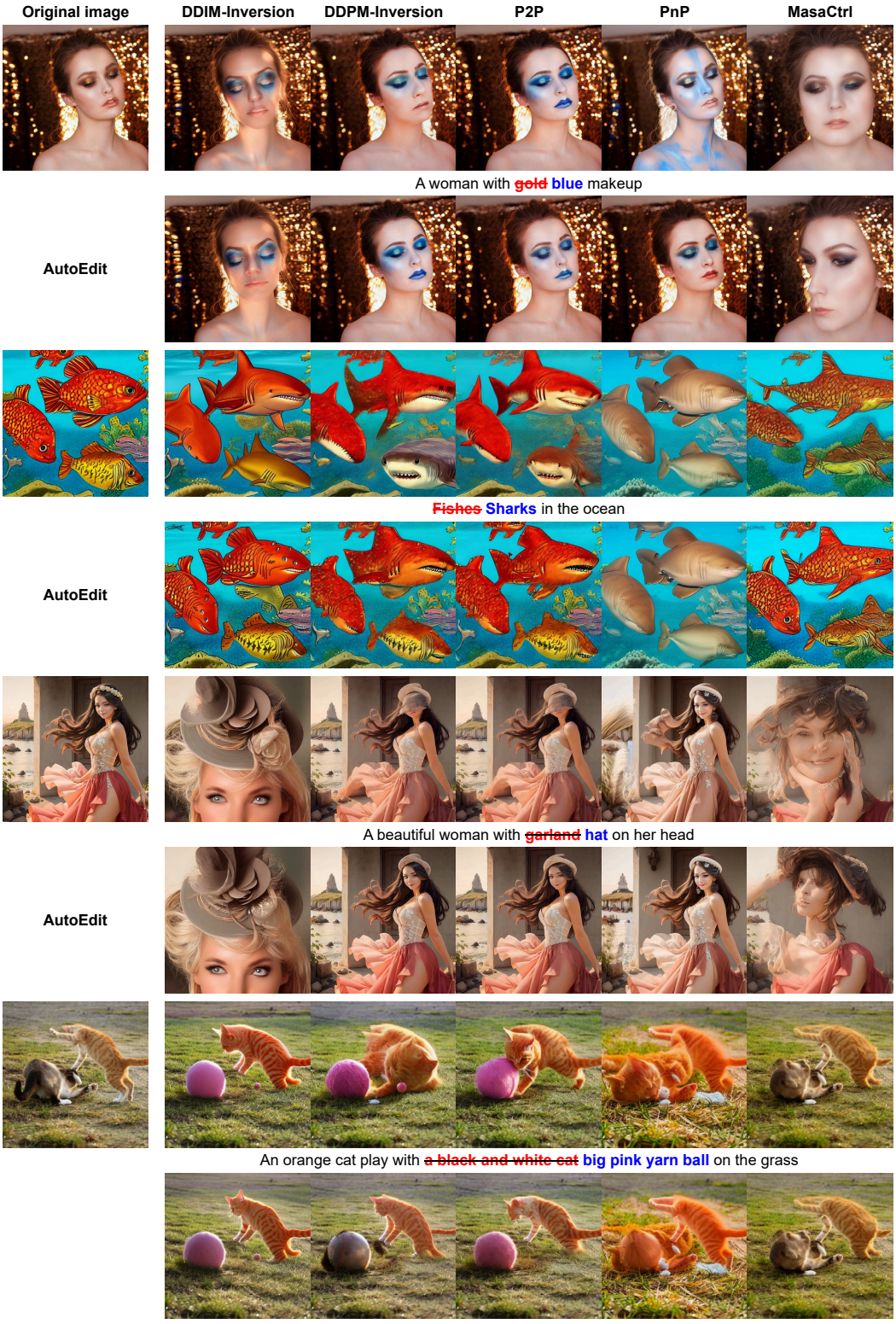


Figure 11: Additional qualitative examples

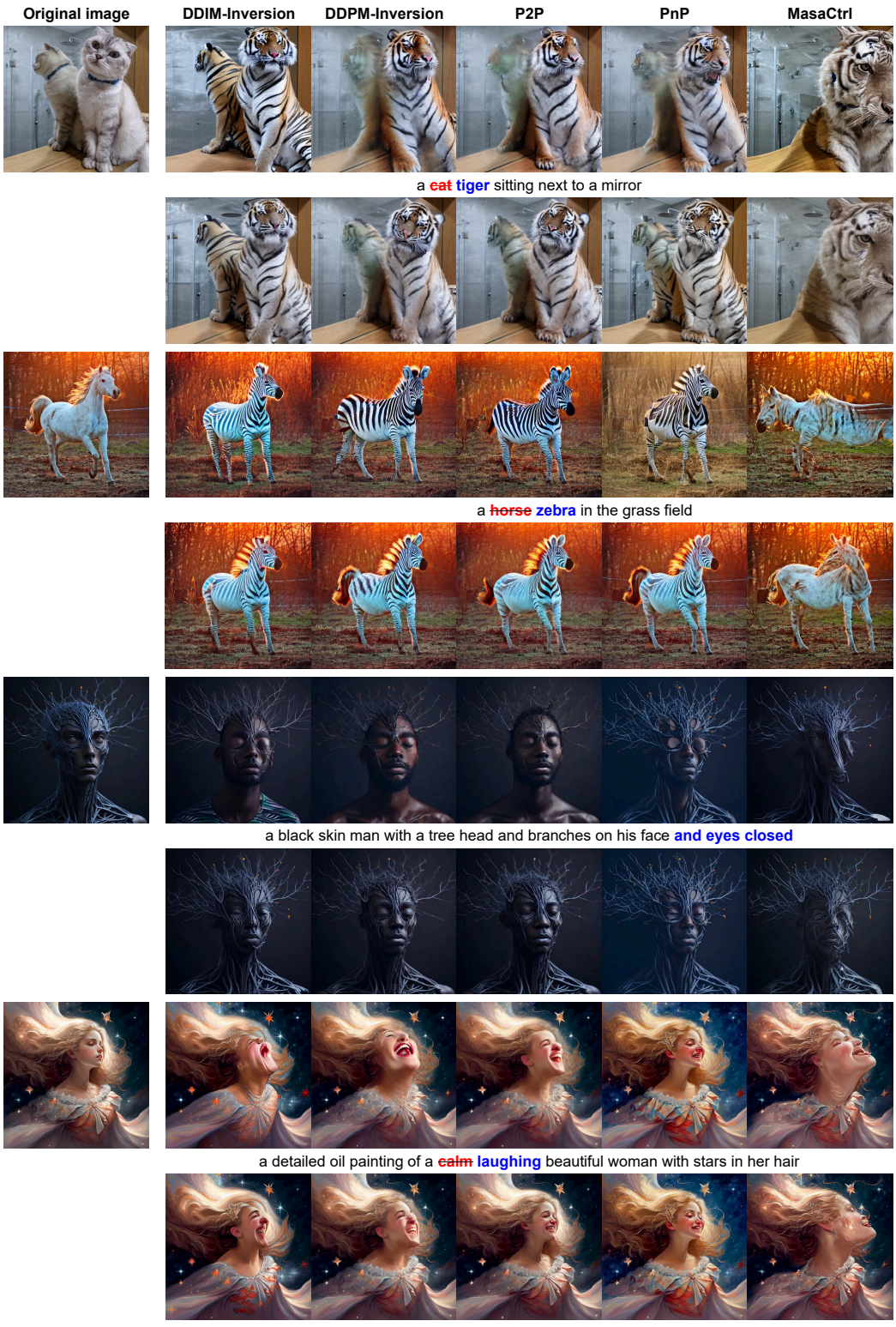


Figure 12: Additional qualitative examples

Figure 13: Qualitative Results on the Flux-based editing. **Top row**: The baseline with the default hyperparameter choice. **Bottom row**: AutoEdit performance. **The prompt** indicates the modification to the original image. This demonstrates the AutoEdit capability to reduce the running required to choose a good hyperparameter.

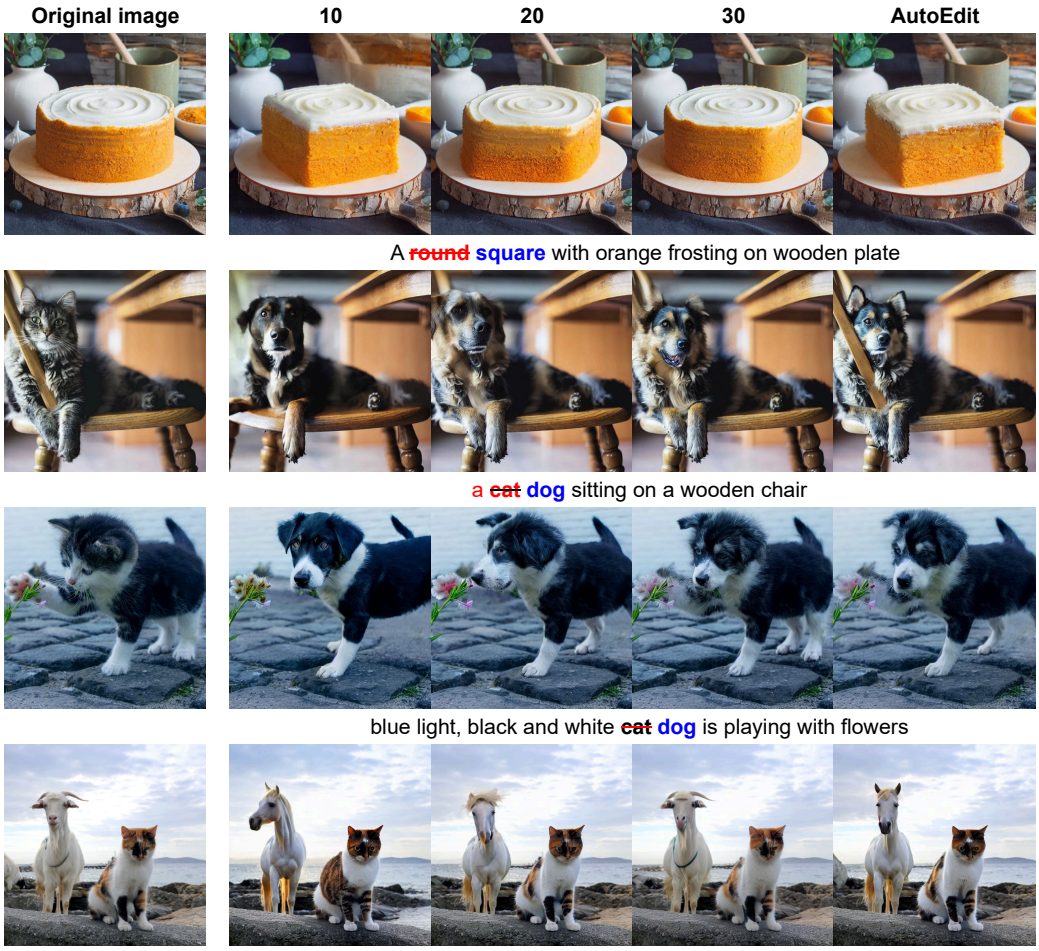


Figure 14: Compare DDPM-Inversion with different values of inversion timestep. In general, AutoEdit can achieve on par or better performance without tuning the inversion timestep

Figure 15: Compare P2P with different values of cross attention ratio. AutoEdit can be on par or better in terms of editing results without needing to tune the hyperparameters

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 provide a comprehensive overview of the background and motivation of this study, effectively outlining its main contributions point-by-point, thus accurately reflecting the paper's scope and significance.

   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: We primarily focused on discussing the limitations associated with this study in the Supplementary Material.

   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: The paper includes the full set of assumptions and correct proofs for each theoretical result which mainly focuses on the complexity of the trial-and-error method and formulating the RL environment.

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: All key contributions of this paper are fully reproducible. We provide detailed implementation and hyperparameter settings for each component in Section 4 and the Supplementary Materials. In addition, we include pseudocode for both training and inference to facilitate understanding and reproducibility of our method.

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: [No]

   Justification: Due to code privacy restrictions, we do not include the source code in the Supplementary Materials. However, our method remains fully reproducible based on the detailed instructions provided in the paper, along with the pseudocode for both the training and testing phases included in the Supplementary Materials.

   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: The paper specifies detailed experimental configurations in Section 4 and more details are provided in Suppelementary, providing readers with essential information to comprehend the results. The setting of the training dataset, and baselines can also be found in the Section 4 and the Supplementary Material.

   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 did not include an analysis of the statistical significance of the experiments, as our method does not rely on specific initialization schemes or fixed random seeds. Moreover, it does not make assumptions such as normally distributed errors.

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: All experiments were carried out on a 2 $\times$ RTX A6000 GPU server.

   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: After carefully reviewing the referenced document, we certify that the research conducted in the paper conforms, in every respect, with the 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: While this paper primarily focuses on the image editing problem, we acknowledge the potential for negative social impacts, such as the creation of misleading or fake images. We strongly encourage users to exercise caution and refrain from using our approach for harmful purposes, including generating DeepFakes or other deceptive content.

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 uses EditBench and PieBench datasets for training and evaluation, which are common and published datasets in the image editing problem.

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: In the paper, we specified the datasets (Pie Bench and Edit Bench) and implementation (Stable Diffusion), and we cite to all of the datasets and the code used in

our implementation. In addition, we also cite other work that we use their code, as well as other related work.

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: [No]

    Justification: We did not include the code in the Supplementary Material. However, the algorithm is reproducible with the instructions from the main paper and the pseudo code from the Supplementary.

    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: We did not conduct any crowdsourcing experiments and 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: We did not conduct any crowdsourcing experiments and 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: [Yes]

Justification: We only employ the LLM to generate the caption for the image. It does not impact the core methodology, scientific rigorousness, or originality of the research and declaration

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.