EDITGRN: INVERSION-FREE IMAGE EDITING BY SOURCE–TARGET GENERATIVE SYNCHRONIZATION

Anonymous authors
Paper under double-blind review

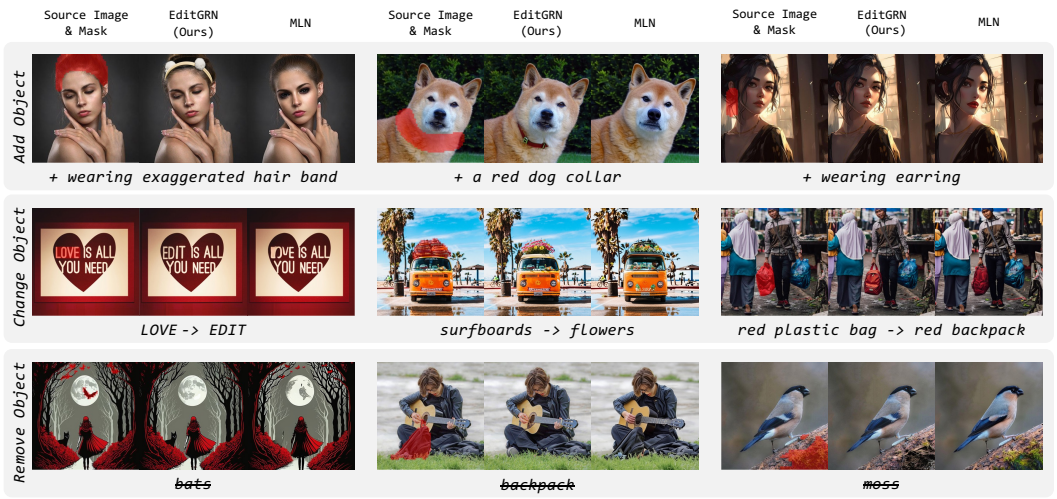


Figure 1: Across object addition, change, and removal, our method, *EditGRN*, produces accurate, contextually coherent edits while preserving unedited content, whereas the state-of-the-art training- and inversion-free discrete method MLN El-Ghoussani et al. (2026) exhibits incomplete modifications and unintended appearance changes in these examples. Red masks indicate editing regions.

ABSTRACT

Text-guided image editing faces a persistent trade-off between preserving source content and realizing coherent target semantics. We explore this challenge through training- and inversion-free image editing based on discrete generative models. Existing inversion-free discrete editors primarily rely on post-hoc probability correction, creating a disconnect between local token-level adjustments and globally coherent semantic generation. To bridge this gap, we introduce *EditGRN*, a training- and inversion-free framework with trajectory-wide source-target generative synchronization in discrete space. *EditGRN* adopts a dual-branch design—the source branch constructs source references and the target branch generates the target image—enabling trajectory-wide synchronization between the two. Exploiting GRN's robustness to random binary corruption, the source branch decouples source-reference construction from any specific generation trajectory by directly corrupting source tokens at varying levels across the full trajectory. The target branch synchronizes with this same corruption schedule, fusing the corresponding source context with editable representations intra-model. By synchronizing source context with target generation along the entire trajectory, our approach faithfully preserves the background while producing high-quality edits that blend seamlessly with the surrounding scene through intra-model fusion. PIE-Bench show state-of-the-art performance on background preservation and text alignment across diverse editing tasks against continuous and discrete baselines. The code will be released.

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

1 INTRODUCTION

Text-guided image editing requires balancing target realization against background preservation. Training-based editors require costly paired editing data (Brooks et al., 2023; Zhang et al., 2023; Fu et al., 2024; Mu et al., 2025), motivating training-free editing on pretrained generators (Hertz et al., 2023; Cao et al., 2023). Discrete models offer a distinct advantage: each token is either retained as source ground truth or independently resampled, providing a directly manipulable interface free from diffusion's entangled per-dimension Gaussian noise.

Image inversion traces generative dynamics backward from a source image to recover a noise trajectory that initializes target-conditioned generation (Ju et al., 2024; Mokady et al., 2023; Wang et al., 2025b). However, recent work shows that explicit inversion is not always essential for effective editing (Xu et al., 2024; Kulikov et al., 2025; Wang et al., 2025c; El-Ghoussani et al., 2026), shifting focus to incorporating source information directly into editing without inversion.

In discrete generation, inversion-free editors such as AREdit (Wang et al., 2025c) and MLN (El-Ghoussani et al., 2026) regulate tokens through post-hoc, source-directed probability correction. Such output-level adjustment leaves a gap between local token nudges and coherent structure generation: as illustrated in Figure 1, it either fails to realize the target edit or entangles source and target features into fragmented, poorly integrated results (e.g., flowers and red backpack). This motivates injecting source context into the model's internal computation rather than correcting probabilities after prediction. How to obtain such source context without inversion in discrete space remains open. Flow-based attempts directly re-corrupt the clean source at each timestep (Zhu et al., 2025), but the resulting states need not lie on a coherent source-generation trajectory, so features extracted from them may mismatch the continuously evolving editing representations.

We propose *EditGRN*, a training- and inversion-free discrete framework built on trajectory-wide source-target generative synchronization. We build on Generative Refinement Networks (GRN) (Han et al., 2026), a discrete generative model trained to recover clean tokens from arbitrarily binary corrupted states rather than trajectory-consistent inputs. This corruption robustness allows the source branch to directly corrupt clean source tokens across the full trajectory, without recovering any particular generation path or accumulating prediction errors. The source and edit branches share the same random corruption schedule, so source context extracted at each step aligns with the evolving editing state throughout the trajectory. Inside the model, this synchronized source context fuses with editable-region representations, simultaneously preserving background fidelity, generating high-quality target content, and yielding smooth edited–unedited integration. Since the edit branch reasons only over editable tokens, its attention cost scales with the edit region rather than the full image—smaller edits run faster, and once source features are extracted they can be reused across multiple edits on the same image. On PIE-Bench, *EditGRN* achieves state-of-the-art background preservation and text alignment. Our contributions are:

- We present *EditGRN*, a training- and inversion-free discrete editing framework built on trajectory-wide source–target synchronization: rather than correcting token probabilities post-hoc, source context is aligned with editing representations throughout refinement and fused intra-model.
- We identify that synchronization arises from sharing the same random corruption schedule across branches. Together with GRN's inherent robustness to binary corruption, it enables trajectory-decoupled source reference construction without inversion or trajectory tracing.
- Extensive experiments on PIE-Bench demonstrate that *EditGRN* achieves state-of-the-art background preservation and text alignment, with computation that adaptively scales with edit-region size.

2 RELATED WORK

Training-based and training-free image editing. Training-based editors learn image-conditioned generation from costly paired or synthetic editing data (Brooks et al., 2023; Zhang et al., 2023; Fu et al., 2024; Mu et al., 2025). Alternatively, pretrained generators can be adapted for editing without task-specific training. Some methods optimize image-specific conditioning embeddings to improve reconstruction (Mokady et al., 2023), while others operate directly at inference through attention

control, feature sharing, or trajectory manipulation (Hertz et al., 2023; Cao et al., 2023; Kulikov et al., 2025; Wang et al., 2025c). Our method requires neither task-specific training nor per-image optimization, using only forward inference with pretrained models.

Inversion-based and inversion-free image editing. Inversion-based methods recover a source noise or latent trajectory before reusing its attention or features during target generation (Hertz et al., 2023; Ju et al., 2024; Wang et al., 2025b; Zhu et al., 2025). Inversion-free alternatives show that source-consistent editing can instead be constructed directly during inference: InfEdit couples source and target diffusion processes (Xu et al., 2024), while FlowEdit defines a direct source-to-target flow (Kulikov et al., 2025). Our method directly re-corrupts encoded source tokens to each GRN refinement level, avoiding recovery of a source trajectory or a continuous noise endpoint.

Discrete generative models for image editing. Discrete editors adapt different generation paradigms: EditAR learns next-token editing (Mu et al., 2025), whereas DICE, VARIN, and EditInfinity recover source representations through discrete inversion or per-image optimization (He et al., 2026; Dao et al., 2025; Wang et al., 2025a). Inversion-free methods instead exploit source–target probability differences, as in AREdit (Wang et al., 2025c), or bias target logits toward source tokens, as in Masked Logit Nudging (El-Ghoussani et al., 2026). Our approach centers on constructing source reference features without inversion. Leveraging GRN's training over random binary-token corruption (Han et al., 2026), we directly corrupt the encoded source at each refinement level, rather than reconstructing its original generation history.

3 Preliminaries: Generative Refinement Networks

Visual Tokenizer. GRN employs a visual tokenizer composed of an encoder $\mathcal{E}$, Hierarchical Binary Quantization (HBQ), and a decoder $\mathcal{D}$. Given an normalized encoder feature $F \in (-1, 1)$, HBQ performs M rounds hierarchical binary quantization:

$$c_i = \sum_{j=1}^{i-1} \frac{\delta(q_j)}{2^j}, \qquad q_i = \begin{cases} 0, & F \le c_i, \\ 1, & F > c_i, \end{cases} \tag{1}$$

where $\delta(0) = -1$ and $\delta(1) = 1$. The quantized feature is reconstructed as $\widehat{F} = \sum_{i=1}^{M} \delta(q_i) 2^{-i}$. Concatenating the binary decisions across feature channels gives the token map

$$z = \mathcal{Q}_{\mathrm{HBQ}}(\mathcal{E}(x)) \in \{0, 1\}^{D \times H \times W}, \tag{2}$$

which can be mapped back to image space by $\mathcal{D}$.

Generative Network. During training, GRN learns to predict the ground-truth token map from a randomly corrupted input:

$$\widetilde{z} = m \odot z + (1 - m) \odot z_{\mathrm{rand}}, \tag{3}$$

where $m \in \{0, 1\}^{D \times H \times W}$ selects ground-truth z or uniformly sampled random bits z_{rand} at each entry. Given $\widetilde{z}$, text condition c, and retention ratio $p = \frac{\|m\|_1}{DHW}$, the Transformer predicts token logits $\ell = f_\theta(\widetilde{z}, c, p)$, from which a token map is sampled. The model is optimized with the cross-entropy

$$\mathcal{L}_{\mathrm{GRN}} = -\frac{1}{DHW} \sum_{k,d} \log p_\theta \left(z_{k,d} \mid \widetilde{z}, c, p \right). \tag{4}$$

By training across randomly sampled retention ratio p, retention maps m, and replacement random binary tokens z_{rand}, GRN learns to recover tokens directly from arbitrary corrupted states, not just from trajectory-consistent inputs. At inference, GRN progressively refines a fully random binary token map by increasing its retention ratio from 0 to 1.

4 Method

4.1 Dual-branch Synchronized Editing Framework

Given a source image x_s, source and target prompts c_s and c_t, and a user-provided mask R, our goal is to modify tokens within R while preserving those in the protected region $\bar{R} = 1 - R$. As

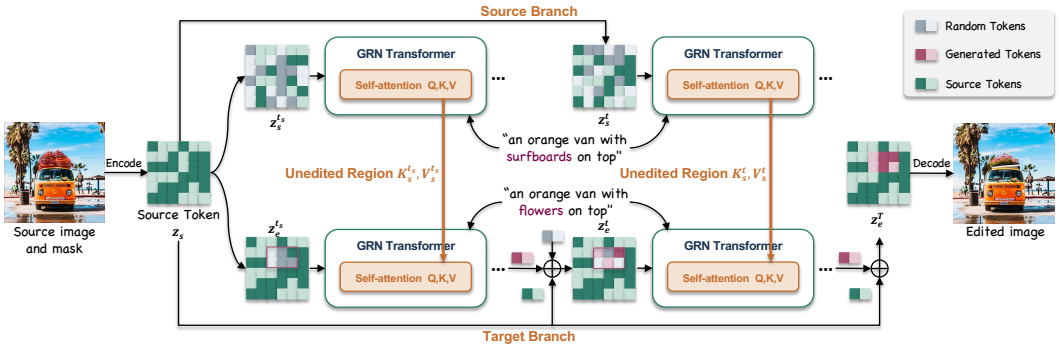


Figure 2: **Overview of EditGRN**. The source branch (top) corrupts clean source tokens at each refinement stage and extracts unedited-region KV; the target branch (bottom) shares the same corruption schedule and replaces its unedited-region KV with the source ones inside every Transformer layer, generating only within the edit region while protected tokens remain anchored to the source.

shown in Figure 2, EditGRN adopts a dual-branch design: the *source branch* constructs source references, and the *target branch* generates the target image. The two branches are synchronized under the same GRN refinement schedule, including retention maps $\{m_t\}_{t=t_s}^{T}$ and random binary tokens z_{rand}, so that corruption intensity is aligned at every step. At each step, the source branch processes a corrupted source state with full self-attention and extracts source keys and values from the visual tokens. Inside every Transformer layer, the target branch replaces its protected region keys and values with these source keys and values, so target queries attend to source-provided context. Predictions are written back only inside R, while protected tokens are restored from source tokens. Algorithm 1 summarizes the procedure.

4.2 Source Reference Construction

Trajectory-decoupled source representations. Exploiting GRN's robustness to random binary corruption, our source branch decouples reference construction from any specific source-generation trajectory. Unlike diffusion or flow models, where continuous Gaussian noise is accumulated along a tightly coupled forward trajectory, GRN corrupts tokens by directly retaining ground-truth bits and replacing the rest with random binary values. Because each corruption step is applied independently to the clean source rather than derived from the previous corrupted state, we can directly construct z_s^t at every stage without tracing any particular generative path:

$$z_s^t = m_t \odot z_s + (1 - m_t) \odot z_{\text{rand}}, \qquad p_s^t = \frac{\|m_t\|_1}{DHW}, \qquad t = t_s, \ldots, T. \tag{5}$$

The retention map m_t only needs to match the scheduled ratio, without requiring any particular relationship to preceding $\{m_t\}$, and z_{rand} is sampled once and shared across steps. Since each z_s^t is anchored to the clean source tokens rather than a preceding source prediction, source prediction errors do not accumulate across steps.

Source context extraction. We next extract source context from these trajectory-decoupled representations for use in the target branch. Motivated by the successful reuse of background KV for guidance in flow-based editing (Zhu et al., 2025), source keys modulate target-query attention over source positions, and source values supply source context for target generation. Specifically, GRN processes the joint sequence $[z_s^t; c_s; p_s^t]$ with full self-attention at each step t, projecting hidden states into queries, keys, and values. We cache the resulting source keys and values $\{K_s^t, V_s^t\}_{t=t_s}^{T}$ corresponding to the visual tokens for all steps. Sharing the same refinement schedule, the source and target branches yield step-synchronized source context for intra-model generation, rather than post-hoc correction of token probabilities. This synchronized source KV guides target-region generation throughout the refinement process, providing consistent context for coherent fusion.

Algorithm 1 EditGRN with trajectory-wide source-target generative synchronization

Require: Source image x_s; source prompt c_s, edited prompt c_e; user-provided mask R; GRN f_θ; retention maps $\{m_t\}_{t=t_s}^{T}$; random binary tokens z_{rand}

$z_s \leftarrow \mathcal{Q}_{\text{HBQ}}(\mathcal{E}(x_s)) \in \{0,1\}^{D\times H\times W}$
$z_s^{t_s} \leftarrow m_{t_s} \odot z_s + (1 - m_{t_s}) \odot z_{\text{rand}}$
$\bar{R} \leftarrow 1 - R$
// Source branch
for $t = t_s, \ldots, T$ **do**
  $p_s^t \leftarrow \frac{\sum_{i,d} m_{t,i,d}}{DHW}$;
  $\{K_s^t, V_s^t\} \leftarrow f_\theta(z_s^t, c_s, p_s^t)$
  $z_s^{t+1} \leftarrow m_{t+1} \odot z_s + (1 - m_{t+1}) \odot z_{\text{rand}}$
end for
// Target branch
$z_e^{t_s} \leftarrow R \odot z_s^{t_s} + \bar{R} \odot z_s$
for $t = t_s, \ldots, T$ **do**
  $p_e^t \leftarrow \frac{\sum_{i,d} R_{i,d} m_{t,i,d}}{\sum_{i,d} R_{i,d}}$
  $\hat{z}_e^t[R] \leftarrow f_\theta\big(z_e^t, c_e, p_e^t; \{K_s^t[\bar{R}], V_s^t[\bar{R}]\}\big)$
  $z_e^{t+1} \leftarrow R \odot [m_{t+1}[R] \odot \hat{z}_e^t[R] + (1 - m_{t+1}[R]) \odot z_{\text{rand}}] + \bar{R} \odot z_s$
end for
return $\widehat{x}_e = \mathcal{D}(z_e^T)$

4.3 Trajectory-Wide Source–Target Generative Synchronization

Trajectory-Wide Synchronization Corruption Schedule. The target branch starts at the same t_s as the source branch and shares the same retention map m_t and random binary tokens z_{rand}. To initialize its state, we take the clean source tokens z_s in the protected region and apply the same corrupted source state $z_s^{t_s}$ inside the edit region:

$$z_e^{t_s} = R \odot z_s^{t_s} + \bar{R} \odot z_s, \qquad p_e^t = \frac{\sum_{i,d} R_{i,d} m_{t,i,d}}{\sum_{i,d} R_{i,d}} \approx p_s^t, \tag{6}$$

where $\bar{R} = 1 - R$. Because m_t selects bits uniformly across the token map, $p_e^t \approx p_s^t$, so the edit region experiences the same corruption intensity as the source branch at every step. This shared corruption schedule is the trajectory-wide synchronization: source context is always extracted at the corruption level matching the current target state, rather than from a fixed clean reference.

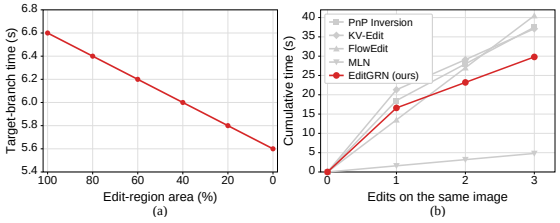


Figure 3: (a) Target-branch time scales with edit-region area. (b) Cumulative time for repeated edits. Measured on an NVIDIA L20 GPU.

Intra-Model Region Fusion. At each Transformer layer, we replace the target branch's protected-region keys and values with the cached source ones at the same step: $\widetilde{K}_e^t = \text{concat}[K_s^t[\bar{R}], K_e^t[R]]$, $\widetilde{V}_e^t = \text{concat}[V_s^t[\bar{R}], V_e^t[R]]$, where $[R]$ and $[\bar{R}]$ index the edit region R and protected region $\bar{R}$, respectively. Target queries in R then attend over this fused visual context together with c_e and p_e^t. After sampling the target prediction $\hat{z}_e^t[R]$ (defined only inside R), we corrupt the edit region for the next step while restoring protected tokens from the clean source:

$$z_e^{t+1}[R] = m_{t+1}[R] \odot \hat{z}_e^t[R] + (1 - m_{t+1}[R]) \odot z_{\text{rand}}[R], \qquad z_e^{t+1}[\bar{R}] = \bar{R} \odot z_s. \tag{7}$$

By fusing synchronized source context with target representations inside the model at every step, the edit region generates with consistent background guidance while protected tokens remain.

Area-Adaptive Editing. The source branch attends over all N_I tokens, while the target branch only computes visual queries for the N_R tokens inside R, so the target-branch cost scales as $\mathcal{O}(N_R N_I)$ rather than $\mathcal{O}(N_I^2)$; as shown in Figure 3 (a), target-branch time decreases linearly as the edit region

Table 1: **Quantitative results on PIE-Bench.** Cont. and Disc. denote continuous (diffusion/flow) and discrete generative models, respectively. Train indicates additional task-specific training, Opt per-image optimization, and Inv explicit image inversion. For SGR-Edit, U and A denote user-provided and attention-derived masks, respectively. CLIP_W and CLIP_E measure alignment over the whole image and edited region.

Method	Type	Train	Opt	Inv	Background Preservation				Text Alignment	
					PSNR↑	$\text{LPIPS}_{\times 10^3}$ ↓	$\text{MSE}_{\times 10^4}$ ↓	$\text{SSIM}_{\times 10^2}$ ↑	CLIP_W ↑	CLIP_E ↑
Additional training or per-image optimization (Train=Y or Opt=Y)										
InstructPix2Pix	Cont.	Y	N	N	16.69	271.33	392.22	68.39	23.49	22.20
MGIE	Cont.	Y	N	N	21.20	142.25	295.11	77.52	24.28	21.79
EditAR	Disc.	Y	N	N	21.32	117.15	130.27	75.13	24.87	21.87
Null-text + P2P	Cont.	N	Y	Y	27.03	60.67	35.86	84.11	24.75	21.86
EditInfinity	Disc.	N	Y	Y	27.95	33.08	24.27	92.12	26.41	23.47
Training-free: continuous generators (Train=N and Opt=N)										
DDIM + P2P	Cont.	N	N	Y	17.87	208.80	219.88	71.14	25.01	22.44
PnP Inversion	Cont.	N	N	Y	22.46	106.06	80.45	79.68	25.41	22.62
InfEdit	Cont.	N	N	N	28.51	47.58	32.09	85.66	25.03	22.22
RF-Edit	Cont.	N	N	Y	24.83	113.15	52.46	83.38	25.57	22.26
KV-Edit	Cont.	N	N	Y	35.87	9.92	4.69	–	22.91	–
FlowEdit	Cont.	N	N	N	21.96	112.19	94.99	83.08	25.25	22.58
Training-free: discrete generators (Train=N and Opt=N)										
DICE	Disc.	N	N	Y	27.29	52.90	43.76	89.79	23.79	21.23
VARIN	Disc.	N	N	Y	26.54	54.04	38.33	85.39	25.05	21.49
AREdit	Disc.	N	N	N	24.19	87.00	–	83.70	25.42	22.77
MLN	Disc.	N	N	N	29.70	36.50	23.30	86.80	26.15	22.72
EditGRN (U)	Disc.	N	N	N	**38.08**	**7.37**	**2.32**	**98.80**	**27.29**	**24.65**
EditGRN (A)	Disc.	N	N	N	33.22	16.87	15.07	97.59	26.80	23.83

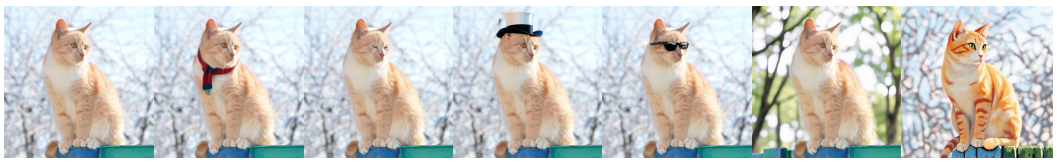


Figure 4: Multiple edits of the same image.

shrinks. This advantage amplifies for repeated edits on the same image (Figure 4). Source features are extracted once and reused, whereas PnP re-inverts on every edit, KV-Edit pays a one-time but expensive inversion, and FlowEdit/MLN pay the full editing cost each time. EditGRN pays only once source once and then the target branch cost per edit, which results in a shallower cumulative-time slope (Figure 3 (b)). While MLN is the fastest, it can result in fragmented, poorly integrated edits in Figure 1. EditGRN retains efficiency without sacrificing quality.

5 Experiments

5.1 Experimental Setup

Comparison Methods. We compare three groups: (1) editors requiring additional training or per-image optimization: InstructPix2Pix (Brooks et al., 2023), MGIE (Fu et al., 2024), EditAR (Mu et al., 2025), Null-text + P2P (Mokady et al., 2023; Hertz et al., 2023), and EditInfinity (Wang et al., 2025a); (2) training-free continuous editors: DDIM + P2P (Hertz et al., 2023), PnP Inversion (Ju et al., 2024), InfEdit (Xu et al., 2024), RF-Edit (Wang et al., 2025b), KV-Edit (Zhu et al., 2025),

and FlowEdit (Kulikov et al., 2025); (3) training-free discrete editors: DICE (He et al., 2026), VARIN (Dao et al., 2025), AREdit (Wang et al., 2025c), and MLN (El-Ghoussani et al., 2026).

Benchmark. PIE-Bench (Ju et al., 2024), a widely adopted benchmark for text-guided image editing. It covers nine editing tasks-object replacement, addition, deletion, and changes to content, pose, color, material, background, and style. The images span natural and artificial scenes across four categories: animals, humans, indoor scenes, and outdoor scenes. Each example provides a source image with its corresponding source prompt, a target editing prompt, and an annotated editing mask.

Metrics. Following PIE-Bench, we report background PSNR, LPIPS, MSE, and SSIM, and CLIP similarity for the whole image and edited region (Caron et al., 2021; Zhang et al., 2018; Wang et al., 2004; Radford et al., 2021).

Implementation Details. We use the pretrained GRN text-to-image model [1]. Experiments runs on a single NVIDIA L20 GPU. By default, we start at $t_s = 14$ with $p_s^{t_s} = 0.10$, retaining approximately 10% of source bits and requiring only $T - t_s = 36$ refinement steps out of $T = 50$.

5.2 Comparison to State-of-the-Arts

Quantitative results. Under the PIE-Bench protocol, EditGRN (U, User-Provided Mask) achieves the best results across all background-preservation and text-alignment, as shown in Table 1. These improvements hold over methods requiring training and training-free continuous editors. Among inversion-free discrete editors, EditGRN also improves both aspects over AREdit and MLN, which employ prediction-probability guidance. Figure 5 further demonstrates strong performance across diverse editing tasks, rather than gains concentrated in a particular task category. Like KV-Edit and EditInfinity, EditGRN (U) uses user-provided mask, while achieving stronger performance than them under the same evaluation protocol.

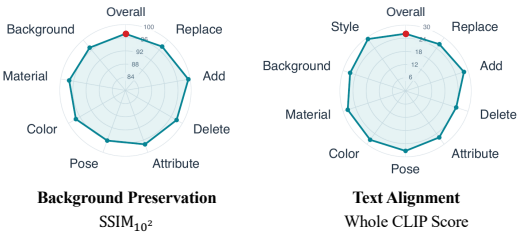


Figure 5: **Per-task results on PIE-Bench.** EditGRN's overall score (red dot) already exceeds other methods, and its per-task scores closely track this overall level across all categories.

Beyond user-provided masks, we also evaluate EditGRN (A, Attention-Derived Mask), which automatically localizes edits using attention maps. Following Prompt-to-Prompt (Hertz et al., 2023) and MLN El-Ghoussani et al. (2026), we identify edited words from source-target prompt differences and extract visual-to-text attention maps from the source branch at $p_s^t = 0.8$. We aggregate visual-to-text attention across subwords to localize edits, reusing the source branch forward pass. Otsu thresholding (Otsu, 1979) then converts the attention maps into binary masks separating editable and background regions. For object addition, we instead use the target prompt in the source branch to extract attention maps for newly added words. EditGRN (A) still achieves competitive performance, suggesting limited reliance on precisely specified editing masks.

Qualitative Results. We compare across all PIE-Bench editing tasks in Figure 6. For space, we include four representative training-free editors: PnP Inversion, an classical inversion-based diffusion method; FlowEdit, an classical inversion-free flow method; KV-Edit, a flow-based inversion method that reuses source background KV; and MLN, a state-of-the-art inversion-free discrete editor. Across diverse tasks, including complex multi-object real-world scenes and text editing in Figure 1, EditGRN preserves unedited content while producing target-aligned modifications.

5.3 Ablation Studies

Ablation studies are performed on the 'random class' of PIE-Bench, covering all types of editing and allowing an efficient and unbiased evaluation.

Ablation on Randomness Robustness. We run EditGRN with shared $\{m_t\}$ and z_{rand} across the two branches but vary the random seeds across independent runs. Table 2 shows that metrics remain

[1] https://huggingface.co/bytedance-research/GRN/blob/main/GRN_T2I_2B_FSA_251600.pth



Figure 6: **Qualitative results on PIE-Bench across all nine tasks.** The red mask denotes the edited region R, expected to follow the target prompt, while other regions retain the background.

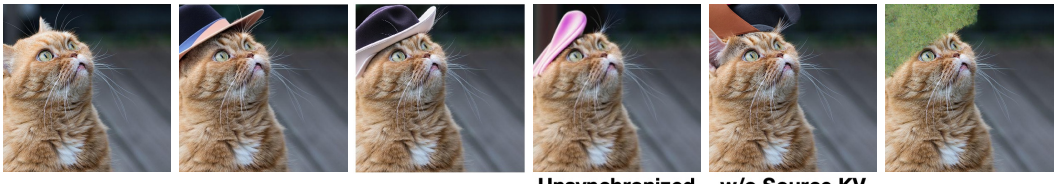


Figure 7: Qualitative results of ablation studies.

stable, and the two runs produce visually consistent edits (Figure 7, EditGRN (1–2)). This supports our trajectory-decoupled design: because GRN recovers clean tokens from arbitrary corrupted states, source references constructed directly from the clean source are robust to random corruption realizations, without needing to recover source-specific generation randomness.

Table 2: Quantitative results of ablating studies.

Variant	Background Fidelity				Text Alignment	
	PSNR↑	LPIPS$_{\times 10^3}$ ↓	MSE$_{\times 10^4}$ ↓	SSIM$_{\times 10^2}$ ↑	CLIP$_{\mathrm{W}}$ ↑	CLIP$_{\mathrm{E}}$ ↑
EditGRN	39.00 ±1e-2	6.78 ±1e-4	2.06 ±3e-6	98.90 ±9e-5	26.39 ±0.05	23.31 ±0.08
Unsynchronized Schedule	38.80	6.74	2.01	98.90	25.96	23.02
w/o Protected Region Writeback	38.29	11.64	2.69	98.56	26.36	23.32
w/o Source KV Injection	38.63	6.56	2.06	98.89	24.72	21.00
Static Source KV	38.30	7.63	2.15	98.84	23.31	19.17
Flow w/ direct source noise	34.71	10.73	6.27	95.72	24.56	22.23

Ablation on Synchronized Corruption Schedule. We then break the shared corruption schedule between the two branches by sampling independent m_t and z_{rand} for the source and target branches. Table 2 shows that text alignment degrades. As shown of Figure 7, the injected source KV no longer matches the target state's corruption level, producing less coherent fusion between the edited hat and the surrounding fur. This confirms that trajectory-wide synchronization, sharing the same corruption schedule across branches, is essential for effective source-guided generation.

Ablation on Protected Region Writeback. We remove the step-wise restoration of protected tokens from the source encoding, while retaining source KV guidance. As shown in Table 2, background fidelity deteriorates. Source KV supplies contextual features but does not enforce equality between protected and source tokens. Token writeback explicitly enforces this constraint, improving background preservation without compromising text alignment.

Ablation on Synchronized Reference. We compare two variants: (1) removing source KV injection, and (2) reusing a static KV cache extracted at the final, least-corrupted source step across all editing steps. Removing source KV degrades text alignment, as shown in Table 2. Figure 7 further suggests that injected source KV provides source-image context for generating objects that remain consistent with the surrounding content. This also clarifies that this variant leaves only the target branch, *i.e.*, bare GRN generation on the edit region, which falls short of our full method. This confirms that the gains do not come from the backbone alone. Static KV further degrades text alignment even compared with no injection, showing that source context alone is insufficient. Its synchronization with the editing branch matters.

Ablation on Trajectory-Decoupled Source in GRN and Flow. To verify that our trajectory-decoupled source construction is specific to GRN's discrete corruption robustness, we port the same strategy to a flow-based editor: following KV-Edit (Zhu et al., 2025), we keep the target branch unchanged but replace its inversion source branch with direct source noise injection via $x_t = (1-t)x_0 + t\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$, and inject the background KV. As shown in Table 2, all metrics degrade sharply. This suggests that in continuous flow space, directly noised states do not lie on a coherent source generation trajectory, and their features mismatch the editing representations. GRN's binary corruption robustness circumvents this limitation in discrete space.

Ablation on Initial Retention Ratio. The initial retention ratio $p_s^{t_s}$ determines how much source information is retained in the edit region at the start of refinement: a larger $p_s^{t_s}$ preserves more source content but leaves less room for target realization. Table 3 and Figure 9 in the Appendix show that this trade-off manifests as improved background fidelity at the cost of weaker text alignment. We choose $p_s^{t_s} = 0.10$ with $t_s = 14$ to balance both objectives.

6 Conclusion

We presented EditGRN, a training- and inversion-free framework for text-guided image editing in discrete space. A dual-branch design with shared corruption schedule synchronizes trajectory-decoupled source references with target generation, fusing source context intra-model to achieve strong background preservation and text alignment without inversion or paired data. Our results suggest that discrete token-level manipulation, combined with synchronized source-target branches, offers a promising direction for efficient and coherent training-free editing.

AI USE STATEMENT

In this work, we used generative AI tools to assist with understanding existing code, diagnosing and fixing implementation bugs, and improving the clarity and grammar of the manuscript. All AI-assisted code changes were manually reviewed and validated through experiments, and all AI-assisted writing was checked and revised by the authors. Generative AI was not used to fabricate data, results, citations, or scientific claims. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18392–18402, 2023.

Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. In *IEEE/CVF International Conference on Computer Vision*, 2023.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *IEEE/CVF International Conference on Computer Vision*, 2021.

Quan Dao, Xiaoxiao He, Ligong Han, Ngan Hoai Nguyen, Amin Heyrani Nobari, Faez Ahmed, Han Zhang, Viet Anh Nguyen, and Dimitris N. Metaxas. Discrete noise inversion for next-scale autoregressive text-based image editing. *arXiv preprint arXiv:2509.01984*, 2025.

Amir El-Ghoussani, Marc Hölle, Gustavo Carneiro, and Vasileios Belagiannis. Prompt-guided image editing with masked logit nudging in visual autoregressive models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Findings*, pp. 4810–4820, 2026.

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. In *International Conference on Learning Representations*, 2024.

Jian Han, Jinlai Liu, Jiahuan Wang, Bingyue Peng, and Zehuan Yuan. Generative refinement networks for visual synthesis. *arXiv preprint arXiv:2604.13030*, 2026.

Xiaoxiao He, Quan Dao, Ligong Han, Song Wen, Minhao Bai, Di Liu, Han Zhang, Felix Juefei-Xu, Chaowei Tan, Bo Liu, Martin Renqiang Min, Kang Li, Faez Ahmed, Akash Srivastava, Hongdong Li, Junzhou Huang, and Dimitris N. Metaxas. Dice: Discrete inversion enabling controllable editing for masked generative models. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 762–772, 2026.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross-attention control. In *International Conference on Learning Representations*, 2023.

Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Pnp inversion: Boosting diffusion-based editing with 3 lines of code. In *International Conference on Learning Representations*, 2024.

Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. In *IEEE/CVF International Conference on Computer Vision*, pp. 19721–19730, 2025.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Jiteng Mu, Nuno Vasconcelos, and Xiaolong Wang. Editar: Unified conditional generation with autoregressive models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7899–7909, 2025.

Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979. doi: 10.1109/TSMC.1979.4310076.

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*, 2021.

Jiahuan Wang, Yuxin Chen, Jun Yu, Guangming Lu, and Wenjie Pei. Editinfinity: Image editing with binary-quantized generative models. In *Advances in Neural Information Processing Systems*, 2025a.

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. In *International Conference on Machine Learning*, pp. 64044–64058, 2025b.

Yufei Wang, Lanqing Guo, Zhihao Li, Jiaxing Huang, Pichao Wang, Bihan Wen, and Jian Wang. Training-free text-guided image editing with visual autoregressive model. In *IEEE/CVF International Conference on Computer Vision*, 2025c.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

Sihan Xu, Yidong Huang, Jiayi Pan, Ziqiao Ma, and Joyce Chai. Inversion-free image editing with language-guided diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. Magicbrush: A manually annotated dataset for instruction-guided image editing. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 36, 2023.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Tianrui Zhu, Shiyi Zhang, Jiawei Shao, and Yansong Tang. Kv-edit: Training-free image editing for precise background preservation. In *IEEE/CVF International Conference on Computer Vision*, 2025.

A Appendix

A.1 Qualitative Comparison with Flow Direct-Noise Ablation

Figure 8 shows representative cases comparing EditGRN against the flow-based variant with direct source noise. In the first row, the flow variant produces visible artifacts when transforming the birds into origami, whereas EditGRN yields cleaner results. In the second and third rows, the flow variant fails to realize the target edit altogether (the horse remains white rather than golden, and the bird remains yellow rather than blue), while EditGRN successfully applies the target attribute. These visual results corroborate the quantitative degradation in Table 2: without a coherent source-generation trajectory, directly noised source features in flow space neither anchor the background reliably nor guide target generation effectively.

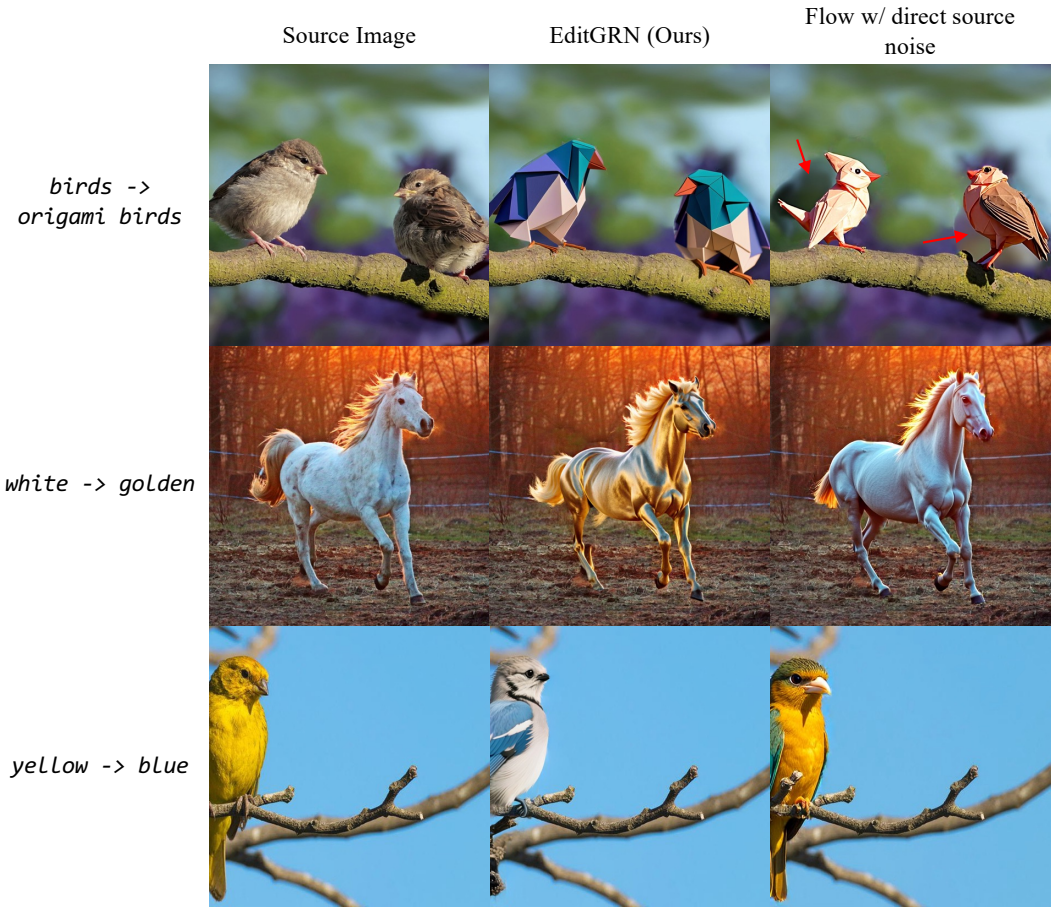


Figure 8: Qualitative comparison of EditGRN against the flow-based variant with direct source noise.

A.2 Extended Analysis of $p_s^{t_s}$ and t_s.

The trade-off in Table 3 arises because $p_s^{t_s}$ directly determines the initial composition of the edit region: when $p_s^{t_s}$ is small, the edit region begins closer to pure random tokens, giving the model full freedom to generate target content but with weaker constraints on how the new object should integrate with the surrounding source scene. As $p_s^{t_s}$ grows, more source tokens are pinned in place, which anchors the edit region to the source layout but also injects source structure that may conflict with the target prompt. Figure 9 illustrates this: with $p_s^{t_s} = 0.20$, the generated pizza bears source-like artifacts that should have been replaced, while with $p_s^{t_s} = 0$ the pizza is cleaner but its shadow and lighting blend less naturally with the table setting. The start step t_s acts on the same axis:

starting refinement later (larger t_s) means the source branch and target branch share fewer, less-corrupted steps, reducing synchronization benefit. Our default $p_s^{t_s} = 0.10$ and $t_s = 14$ gives the balance between target realization and scene integration.

Table 3: Ablation on initial retention ratio $p_s^{t_s}$ and start step t_s.

		Background Fidelity				Text Alignment	
$p_s^{t_s}$	t_s	PSNR↑	LPIPS$_{\times 10^3}$ ↓	MSE$_{\times 10^4}$ ↓	SSIM$_{\times 10^2}$ ↑	CLIP$_{\text{W}}$ ↑	CLIP$_{\text{E}}$ ↑
0.00	0	38.49	7.30	2.30	98.86	26.59	23.67
0.10	14	38.95	6.73	2.06	98.90	26.34	23.28
0.20	20	39.72	6.26	1.80	98.95	25.83	22.88



Figure 9: Qualitative comparison of different initial retention ratio and start step.

A.3 Additional Qualitative Results.

Figure 10 presents more editing results across diverse tasks. EditGRN successfully realizes the target edits while faithfully preserving the unedited regions. As highlighted in the zoomed-in views, the boundaries between edited and preserved regions blend naturally, with consistent lighting, texture, and shadow-confirming the effectiveness of intra-model source context fusion.

A.4 Limitations

This work focuses on static image editing; its extension to video and temporal consistency remain unexplored. Applying our method independently to each frame does not explicitly enforce cross-frame consistency and may introduce appearance variations or temporal flickering. Motion and occlusion further alter the editing region and its surrounding context, making frame-wise source guidance insufficient to guarantee temporally coherent edits. Future work should investigate temporally consistent source-feature construction and edit-region tracking, while assessing the computational and memory costs of caching features across frames.

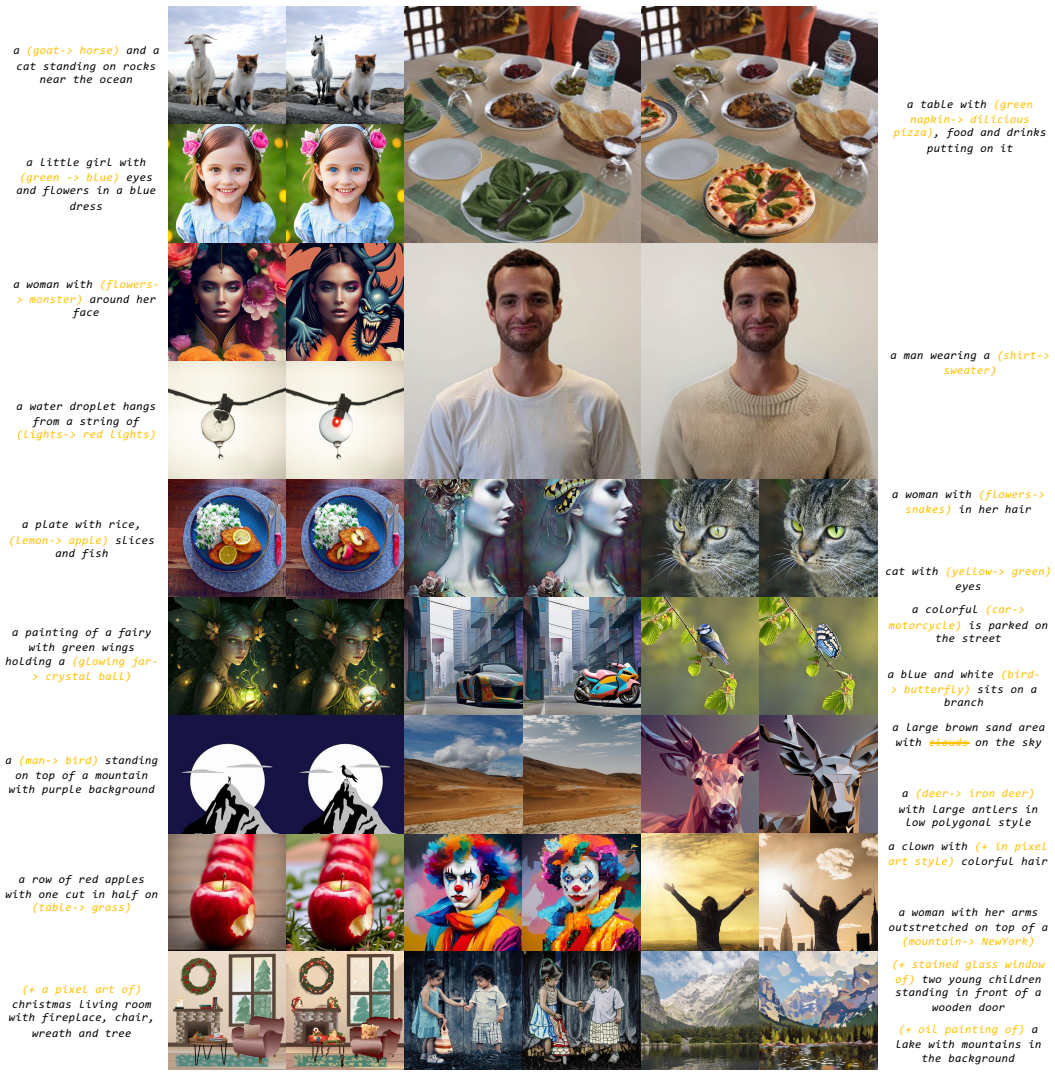


Figure 10: **Additional editing results.** For each pair, the left is the source image and the right is EditGRN's output. The zoomed-in views highlight smooth boundary blending between edited and preserved regions.