CONTROLEDIT: IMAGE EDITING WITH UNPAIRED POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent instruction-based editing models achieve impressive editing performance by fine-tuning pretrained text-to-image models with large-scale pair (source image, target image) editing data. However, constructing such pair editing data is expensive, often requiring a complex data pipeline and human verification, which makes the process of finetuning text-to-image model to image editing model costly and difficult. In this work, we propose **ControlEdit**, a post-training framework that converts a pretrained text-to-image model into an instruction-based image editing model without requiring target image during training. Our framework consists of two stages. The first stage converts text-to-image diffusion model to take instruction and source image as editing condition, while the second stage applies reinforcement learning post-training to further optimize editing quality with respect to prompt alignment and background preservation. Experiments across multiple editing tasks demonstrate that **ControlEdit** achieves competitive performance with supervised editing methods while eliminating the need for paired edited images.

1 INTRODUCTION

Recent advances in pretrained text-to-image generation (Rombach et al., 2021; Chen et al., 2023; Esser et al., 2024; Podell et al., 2024; Labs et al., 2025) have enabled the applications in various domain (Brooks et al., 2023; Zhang et al., 2023; Ruiz et al., 2023; Lugmayr et al., 2022; Poole et al., 2022). Within these domains, image editing (Brooks et al., 2023; Hertz et al., 2022; Wu et al., 2025; Liu et al., 2025) has emerged as a critical subfield that addresses the problem of modifying an original image satisfying two criteria: prompt alignment and background preservation. The first attempts leverage inversion-based methods (Cao et al., 2023; Hertz et al., 2022; Deng et al., 2024; Parmar et al., 2023) to conduct the image editing through a two-stage approach. Firstly, the image is inverted to extract a list of noises by an inversion method (Huberman-Spiegelglas et al., 2024; Mokady et al., 2023; Deng et al., 2024; Dao et al., 2025), which contains the bias information of the source image. The second stage applies the editing operation by directly modifying the list of inverted noises to steer the generated image towards editing instruction. By choosing appropriate hyperparameters, inversion-based image editing can balance between prompt alignment and background preservation. Despite their simplicity and low cost, inversion-based editing methods have several limitations. First, they typically rely on pair of source and target prompts, which may not precisely express the desired edit intention. Second, their performance often lag behinds the training-based method. In contrast, instruction-based editing technique provides directly base on the editing instruction. Recent works (Brooks et al., 2023; Liu et al., 2025; Wu et al., 2025; Labs et al., 2025) finetune pretrained diffusion models to take source image and instruction as editing condition to generate edited image, demonstrating strong performance across diverse editing tasks.

A key limitation of these instruction-based editing techniques is the cost of constructing pair of training data including source and target image. The target images are typically generated by using exiting image editing technique Brooks et al. (2023) or complex editing pipeline Liu et al. (2025), then followed by human verification to ensure editing quality. Therefore, constructing such large-scale, high-quality paired editing data could be extremely resource-consuming and time-consuming. This limitation motivates us to explore unpaired editing approaches that adapt pretrained text-to-image models without requiring target edited images. In this work, we formulate image editing as a conditional generation problem: given a target prompt, the model must synthesize a target image while

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.

remaining anchored to the original source image. From this perspective, editing can be viewed as generation under constraints imposed by the source image. We find that guiding generation toward the target prompt while incorporating source-image information through a ControlNet-based mechanism gives rise to an emergent image-editing capability. To further enhance the editing capability, we perform RL post-training for image editing, optimizing two key criteria: prompt alignment and background preservation. Since existing post-training methods typically assume an already capable base model, **ControlEdit** adopts a two-stage training strategy. The first stage equips the pretrained text-to-image model with basic editing capability, while the second stage applies RL post-training to further improve editing quality.

In summary, our contributions are:

1. We identify an emergent image-editing capability of ControlNet when conditioned on a source image and an edit prompt, showing that source-image conditioning can enable meaningful editing without direct supervision from target edited images.
2. We introduce **ControlEdit** a post-training framework that adapts pretrained text-to-image models for image editing without requiring paired edited images.
3. We demonstrate that **ControlEdit** achieves competitive performance with its supervised counterpart while eliminating the need for target edited images during training.

2 Related work

Text-based image editing. Text-based image editing methods can be broadly categorized into inversion-based and instruction-based approaches. Inversion-based methods (Hertz et al., 2022; Cao et al., 2023; Deng et al., 2024; Huberman-Spiegelglas et al., 2024; Dao et al., 2025) first invert the source image into a noisy latent representation and then apply editing operations, such as attention or feature manipulation, to steer the denoising trajectory toward the desired edit. A key component of these methods is the inversion mechanism itself, and several approaches (Song et al., 2022; Huberman-Spiegelglas et al., 2024; Wallace et al., 2023; Deng et al., 2024) have been proposed to improve reconstruction accuracy and efficiency. DDPM Inversion (Huberman-Spiegelglas et al., 2024) records a sequence of noise residuals during inversion to enable exact source-image reconstruction, while FireFlow (Deng et al., 2024) introduces a specialized solver for rectified-flow inversion that approximates second-order accuracy while retaining the efficiency of a first-order Euler update. In contrast, instruction-based image editing methods (Brooks et al., 2023; Wu et al., 2025; Liu et al., 2025; Zhao et al., 2024; Labs et al., 2025) directly modify a source image according to a natural-language editing instruction. These methods are typically trained on paired source and target images with corresponding editing instructions, learning to generate the target image conditioned jointly on the source image and the instruction. Recent approaches increasingly scale model capacity, training data, and instruction diversity to support a wide range of edits, including object manipulation, style transfer, background modification, and compositional changes. However, their performance remains strongly dependent on the quality and coverage of paired editing data, making large-scale data construction a central challenge for this paradigm.

Several recent works have explored training image editing models without paired source–target supervision (Kumari et al., 2025; Tewel et al., 2026). NP-Edit (Kumari et al., 2025) directly optimizes the diffusion Transformer using feedback from a vision-language model (VLM), which introduces substantial computational overhead during training. Bootstrap Generator (Tewel et al., 2026), in contrast, employs a cycle-consistency objective that reconstructs the source image from the edited output. However, because this objective does not directly assess the quality of the intermediate edited image, it may lead to suboptimal editing behavior.

RL post-training for image generation Previous RL-based post-training methods improve text-to-image generation with respect to specific objectives, such as text-image alignment and visual aesthetics, using either off-policy (Wallace et al., 2024) or on-policy optimization (Liu et al., 2026; Zheng et al., 2026). The reward can be rule-based, e.g., GenEval (Ghosh et al., 2023), or model-based, e.g., PickScore (Kirstain et al., 2023) and ImageReward (Xu et al., 2023). RL has also been applied to optimize prompts (Li et al., 2024; Mo et al., 2024) for text-to-image generation and image editing hyperparameters (Pham et al., 2026). NP Edit (Kumari et al., 2025) directly backpropagates a VLM-based score through the VLM to update the parameters of the image editing model, which

introduces additional computational and implementation overhead. In contrast, our method uses the VLM only as a non-differentiable reward function, allowing it to be seamlessly integrated into standard RL post-training frameworks without modifying the post-training pipeline.

3 PRELIMINARIES

Diffusion models. Diffusion and flow-based generative models (Ho et al., 2020; Lipman et al., 2022) generate images by learning to transform noisy samples into clean data. These models generally involve two processes. (1) The *forward process* progressively perturbs a clean sample $x_0 \sim p_{\text{data}}$ toward noise, with the marginal distribution typically written as $x_t = \alpha_t x_0 + \sigma_t \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$, $t \in [0, 1]$, and α_t and σ_t are determined by the noise schedule, such that $x_1 \sim \mathcal{N}(0, I)$. (2) The *reverse process* learns to transform noisy samples back toward the data distribution, typically by predicting either the noise ϵ (Ho et al., 2020) or a velocity field v (Lipman et al., 2022). For instance, Flow Matching trains a velocity model v_θ with the objective

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{x_0,t,\epsilon}\left[\|v - v_\theta(x_t, t)\|_2^2\right], \tag{1}$$

where, under a linear probability path, $x_t = (1-t)x_0 + t\epsilon$ and the corresponding target velocity is $v = \epsilon - x_0$.

Instruction editing. Image editing aims to transform a source image I^{src} into an edited image I^{edit} while satisfying two key criteria: (1) *Prompt alignment*: the edited image should faithfully follow the editing instruction specified by the instruction prompt p_{instr}; and (2) *Background preservation*: regions and content unrelated to the intended edit should remain consistent with the source image. In some editing settings, a source prompt p_{src} describing the original image and a target prompt p_{edit} describing the desired edited image may also be provided.

Reinforcement learning for image generation. Recent reinforcement learning approaches for image generation (Liu et al., 2026; Zheng et al., 2026) have shown that post-training with task-specific rewards can improve generation quality along targeted criteria. DiffusionNFT (Zheng et al., 2026) performs reinforcement learning through the forward diffusion process rather than optimizing likelihoods along the reverse sampling trajectory. Given samples generated by the old policy v_{old}, it normalize the rewards into the probability $r(x_0, c) \in [0, 1]$ which softly determines its contribution to the positive and negative objectives. Diffusion NFT optimizes the following training objective:

$$\mathcal{L}_{\text{NFT}} = r\left\|v_\theta^+(x_t, c, t) - v\right\|_2^2 + (1-r)\left\|v_\theta^-(x_t, c, t) - v\right\|_2^2, \tag{2}$$

where v is the standard flow-matching velocity target and v_θ^+ and v_θ^- are implicit positive and negative policies. By weighting the positive and negative objectives according to the reward, this formulation implicitly steers the learned velocity field toward high-reward generations and away from low-reward ones, thereby defining a policy-improvement direction.

4 METHOD

In this section, we first demonstrate that image-editing capability naturally emerges from ControlNet pretraining, even when ControlNet is trained solely to reconstruct the source image. Building on this observation, we introduce **ControlEdit** a two-stage training framework that transforms this emergent capability into a strong instruction-based image editor. In the first stage 4.1, we train ControlNet with a source-image reconstruction objective, establishing the initial editing capability without requiring paired editing data. In the second stage 4.2, we jointly leverage the pretrained ControlNet and diffusion Transformer for post-training, further enhancing editing quality in terms of both target-prompt alignment and source-image preservation.

4.1 DISENTANGLE EDITING AND PRESERVATION: TRAINING THE CONTROLNET

To convert a pretrained text-to-image model into an image-to-image editing model, the source image must be encoded and incorporated into the generative backbone. We propose to disentangle the editing objective into two complementary components: *background preservation* and *prompt alignment*. Specifically, we view the pretrained text-to-image diffusion model as the prompt-alignment module,

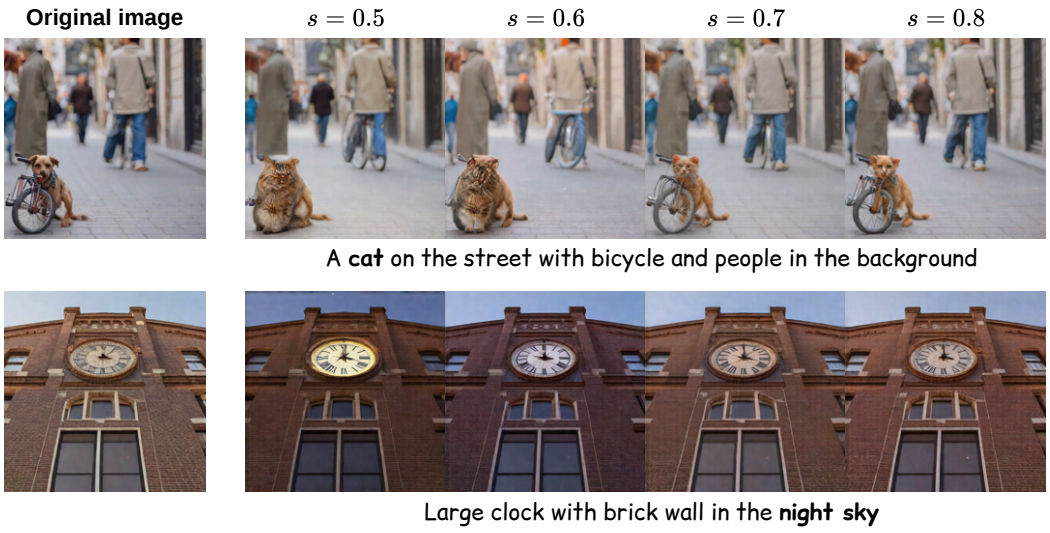

Figure 1: Effect of the conditioning scale s on image editing quality. Different values of s produce substantially different editing outcomes, illustrating the importance of selecting an appropriate conditioning strength for each input.

which drives generation toward the target text prompt, while ControlNet (Zhang et al., 2023) serves as the background-preservation module by conditioning the generation on the source image. By injecting ControlNet features into the pretrained text-to-image diffusion model, the resulting model acquires an image-editing capability: it generates an output aligned with the target prompt while remaining anchored to the source image through the ControlNet features.

More specifically, ControlNet consists of a lightweight stack of Transformer blocks that operates in parallel with the main generative Transformer. The source-image representation is incorporated into the main Transformer through residual feature injection:

$$v_\theta^i \leftarrow v_\theta^i + sC_\tau^j, \tag{3}$$

where C_τ^j denotes the residual feature produced by the j-th ControlNet block corresponding to the i-th block of the main Transformer, and s is a conditioning scale that controls the strength of the injected source-image signal. This conditioning scale provides a direct mechanism for controlling the degree of source-content preservation: a larger s enforces stronger adherence to the source image, while a smaller s permits greater deviation from the original content in favor of the target prompt. We denote the ControlNet residual conditioned on the noisy latent x_t at timestep t and the source latent x_0^{src} as $C_\tau(x_t, t, x_0^{\text{src}})$. Thus, the ControlNet conditioning scale s naturally exposes a controllable trade-off between source-image preservation and target-prompt-driven modification, providing a simple interface for adapting a pretrained text-to-image model to image editing.

ControlNet Training. In the first stage, we train the ControlNet parameter τ by using standard flow matching loss to reconstruct the original image:

$$\hat{\tau} = \arg\min_\tau \mathbb{E}_{x_0^{\text{src}},t}[\|v_\theta(x_t, t, p_{\text{src}}, C_\tau(x_t, t, x_0^{\text{src}}), s = 1) - v\|^2] \tag{4}$$

where $v = \epsilon - x_0^{\text{src}}$. This process can be easily overfitted, thus, we only train this for 2k iterations with batch size of 32.

Edit with target prompt. Given the ControlNet features extracted from the source image under the source prompt p_{src}, we inject these features into the text-to-image generation process conditioned on the target prompt p_{target}. This enables the model to generate an edited image that aligns with the target prompt while preserving content from the source image. Figure 1 illustrates the editing results obtained with different conditioning scales s. We observe that the optimal conditioning strength varies across editing tasks. Edits involving substantial changes to the source image, such as style transfer or background modification, generally benefit from a smaller s, allowing the generative model greater flexibility to deviate from the source content. In contrast, more localized edits that require stronger preservation of the source image, such as object addition or object swapping, typically favor a larger s. These observations highlight the importance of selecting an appropriate

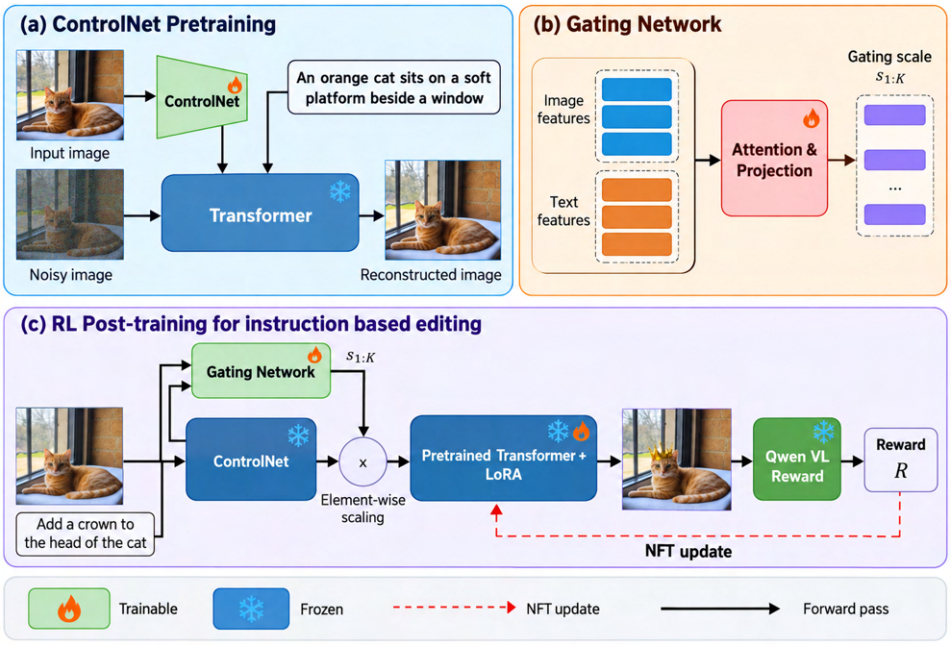

Figure 2: Overview of **ControlEdit**(a) ControlNet pretraining for initializing and training the content-preservation module. (b) Gating network for predicting adaptive conditioning scales. (c) RL post-training for further fine-tuning the pretrained text-to-image model into an instruction-based image editing model.

conditioning scale s to balance target-prompt alignment and source-image preservation for different editing tasks. To adapt this balance to different editing instructions, we introduce a gating network G_ϕ that predicts an adaptive value of s.

Warm-up instruction distillation. As shown above, the pretrained text-to-image model with controlnet naturally supports target-prompt-based editing, where the target prompt p_{edit} directly describes the desired edited image. However, practical image editing is typically specified through an editing instruction p_{instr}, which describes how the source image should be modified rather than what the final image should look like. We therefore aim to transfer the model's existing target-prompt editing capability to instruction-conditioned editing. To this end, we adopt the distillation strategy from Bootstrap Generator (Tewel et al., 2026), using the target-prompt-conditioned prediction as a teacher for the instruction-conditioned prediction. Specifically, we optimize

$$\mathcal{L}_{\text{distill}} = 1 - \cos\left(v_{\text{instr}} - v_{\text{src}}, v_{\text{edit}} - v_{\text{src}}\right) + \lambda \left| v_{\text{instr}} - v_{\text{edit}} \right|_2^2, \tag{5}$$

where v_{src}, v_{edit}, and v_{instr} denote the model predictions conditioned on the source prompt p_{src}, target prompt p_{edit}, and editing instruction p_{instr}, respectively. Only the newly introduced LoRA parameters for instruction conditioning are optimized. The directional term encourages p_{instr} to induce the same source-to-target transformation as p_{edit}, while the ℓ_2 term directly aligns their predictions. In this way, we distill the model's target-prompt editing capability into instruction-based editing, providing an effective initialization for subsequent training.

4.2 Post-training: Turning text-to-image to image editing model

In this section, we introduce an RL-based training mechanism that jointly optimizes the gating network G_ϕ to adaptively predict the conditioning scale s and fine-tunes the main transformer for fine-grained instruction-based image editing in Figure 2c.

Gating network design. We design the gating network G_ϕ to adaptively determine the conditioning strength based on both the source image and the editing instruction p_{instr}. Specifically, G_ϕ takes as input the layer-wise ControlNet features together with the instruction embeddings. We employ a layer-wise attention module to capture the interaction between the source-image features and the instruction semantics, allowing the network to infer how strongly the source content should

be preserved for a given edit. The attended features are subsequently pooled and passed through a projection layer to predict layer-specific conditioning scales. For a ControlNet with K layers, the gating network produces

$$G_\phi(\cdot, p_{\text{instr}}) = \{s_1, s_2, \ldots, s_K\}, \tag{6}$$

where s_k controls the strength of the ControlNet residual injected at the k-th layer of the generative Transformer. This enables the model to adapt the source-conditioning strength across both editing instructions and network layers, rather than relying on a single manually selected global scale. Further architectural and training details are provided in Appendix A.4.

Reward function. We design the reward function to jointly evaluate two key editing objectives: *background preservation* and *instruction alignment*. Following NPEdit (Kumari et al., 2025), we employ a pretrained vision-language model (VLM), Qwen3VL-4B-Instruct (Team, 2025a), to assess the edited image with respect to both criteria. To enable efficient reward evaluation during RL post-training, we operate the VLM in a forward-scoring mode, avoiding the computational overhead of autoregressive generation. Further details on the reward function are provided in Appendix A.3.

RL post-training. In the second stage, we apply reinforcement learning (RL) post-training to further enhance the model's editing capability. We adopt DiffusionNFT (Zheng et al., 2026) to jointly optimize the gating network G_ϕ and the LoRA of transformer v_θ. Given the layer-wise conditioning scales $s_1, s_2, \ldots, s_K$ predicted by G_ϕ, we perform rollouts to generate G candidate images for each image-instruction pair and compute a reward for each generated image. Following DiffusionNFT, the training objective is constructed using positive and negative predictions:

$$v^+ = \beta v_\theta + (1-\beta) v_{\theta_{\text{old}}}, \qquad v^- = (1+\beta) v_{\theta_{\text{old}}} - \beta v_\theta, \tag{7}$$

where θ_{old} denotes the rollout model. The corresponding positive and negative objectives are

$$\mathcal{L}^+ = \mathbb{E}\left[\|v^+ - v\|_2^2\right], \qquad \mathcal{L}^- = \mathbb{E}\left[\|v^- - v\|_2^2\right]. \tag{8}$$

where $v = \epsilon - x_0$ is the target velocity. Given the normalized reward $r \in [0, 1]$, the training loss interpolates between the positive and negative objectives as:

$$\mathcal{L}_{\text{policy}} = r\mathcal{L}^+ + (1-r)\mathcal{L}^-. \tag{9}$$

Thus, samples receiving higher rewards place greater weight on the positive objective, whereas lower-reward samples are increasingly optimized toward the negative objective. To prevent the post-trained model from deviating excessively from the pretrained editing model, we further introduce a reference-model regularization term:

$$\mathcal{L}_{\text{reg}} = \|v_\theta - v_{\text{ref}}\|_2^2, \tag{10}$$

where v_{ref} is the velocity prediction of the frozen reference model obtained after the warm-up instruction-distillation stage, with the conditioning scale fixed to $G_\phi = 1$. The final training objective is

$$\mathcal{L} = \mathcal{L}_{\text{policy}} + \lambda \mathcal{L}_{\text{reg}}, \tag{11}$$

where we set $\lambda = 0.01$. The rollout parameters are updated using an exponential moving average (EMA) of the online model:

$$\theta_{\text{old}} \leftarrow \gamma \theta_{\text{old}} + (1-\gamma)\theta, \tag{12}$$

with $\gamma = 0.5$. We provide the full pseudocode is given in Appendix A.5.

5 Experiments

5.1 Experimental setup

Training details. In the first stage, we train the ControlNet with a source-image reconstruction objective, conditioned on the corresponding source prompt p_{src}. We use a learning rate of 5×10^{-5} and train on 4 GPUs with a per-GPU batch size of 8. Since the objective of this stage is simply to establish source-image reconstruction capability, we find that a short training schedule is sufficient and train the ControlNet for only 2K iterations. We subsequently perform instruction-distillation warm-up for 3K iterations using the same learning rate and batch size.

Method	Base Model	Training	#params	SC↑	PQ↑	Overall ↑
Step1x Edit (Liu et al., 2025)	Flux	Supervised	12B	7.61	7.12	7.16
Qwen ImageEdit (Wu et al., 2025)	-	Supervised	20B	6.97	7.49	6.97
Flux-Kontext (Labs et al., 2025)	Flux	Supervised	12B	6.83	7.15	6.49
NPEdit (Kumari et al., 2025)	Z-Image	Unsupervised	6B	5.65	7.36	5.50
ControlEdit-M	SD3-medium	Unsupervised	2B	6.12	7.17	6.01
Supervised	SD3-medium	Supervised	2B	5.84	6.45	5.37
ControlEdit-L	SD3-large	Unsupervised	8B	6.32	7.38	6.23
Supervised	SD3-large	Supervised	8B	6.24	6.95	6.09

Table 1: Quantitative results of **ControlEdit** with other baselines

In the second stage, we perform RL post-training to further enhance the editing capability. During this stage, we jointly optimize the gating network G_ϕ, which predicts adaptive conditioning scales for ControlNet feature injection, and LoRA parameters of the main diffusion Transformer to improve editing fidelity. We set the LoRA rank to 128 and otherwise follow the LoRA configuration used in the original implementations of Diffusion NFT and FlowGRPO. For the gating network G_ϕ, we employ layer-specific query, key, and value projections to model the interaction between the ControlNet features and instruction embeddings. The resulting features are pooled and passed through a two-layer MLP with SiLU activations, followed by a sigmoid function that constrains each predicted conditioning scale to $(0, 1)$. We perform RL post-training with Diffusion NFT for 30K iterations. During each RL rollout, for every source-image–instruction pair $(x_0^{\mathrm{src}}, p_{\mathrm{instr}})$, we sample 16 candidate edited images using the standard flow-matching denoising scheduler. The generated samples are evaluated by a pretrained Qwen3-VL-4B model to obtain rewards for RL optimization. Unless otherwise specified, all experiments use SD3-Medium as the base text-to-image model. All training is conducted on 4 NVIDIA A100 40GB GPUs. Additional implementation and training details are provided in Appendix A.4.

Dataset. To train **ControlEdit** we use the GPT Image 1.5M dataset (Wang et al., 2025). Since a substantial portion of the dataset does not include the source prompts corresponding to the original images, we employ Qwen3-VL-4B to generate captions for the source images. These generated captions serve as the source prompts p_{src} during the first-stage ControlNet training.

Baselines. Since the GPT Image 1.5M dataset provides paired edited images, we construct a supervised counterpart of **ControlEdit** for a controlled comparison. The supervised baseline uses the same pretrained backbone and model architecture, including the ControlNet and Transformer LoRA modules, but jointly optimizes the ControlNet and LoRA parameters using the synthesized target images as direct supervision. This comparison allows us to assess the effectiveness of our training strategy against supervised training under the same model architecture and training data.

We further compare **ControlEdit** with several representative image editing methods, including NP Edit (Kumari et al., 2025), Qwen Image Editing (Wu et al., 2025), Step1x Edit (Liu et al., 2025), and FLUX.1 Kontext (Labs et al., 2025). For NPEdit, we use the authors' released checkpoint based on Z-Image (Team, 2025b). Unless otherwise specified, we use 10 sampling steps for all baselines and **ControlEdit** except for NP Edit, for which we follow the sampling config of the released checkpoint.

Metrics. We evaluate **ControlEdit** and all competing baselines on GEdit-Bench (Wang et al., 2024). Following prior work (Kumari et al., 2025; Tewel et al., 2026), we adopt VIEScore (Ku et al., 2024) with Qwen-VL-72B (Team, 2025a) as the primary metric for evaluating image editing performance. VIEScore assesses two complementary aspects of editing quality: (1) *Semantic Consistency (SC)*, which measures how well the edited image satisfies the target editing instruction, and (2) *Perceptual Quality (PQ)*, which evaluates the visual quality of the resulting image, including realism and the presence of artifacts. The overall VIEScore is computed as the geometric mean of SC and PQ, jointly reflecting instruction adherence and perceptual quality.

5.2 Main results

Quantitative Analysis. We present the main results of **ControlEdit** in Table 1. We instantiate **ControlEdit** with SD3-Medium and SD3-Large backbones, denoted as **ControlEdit-M** and **ControlEdit-L**, respectively. Compared with their corresponding supervised ControlNet baselines, both



Figure 3: Qualitative comparison of **ControlEdit** with other baselines.

variants of **ControlEdit** consistently achieve higher SC, PQ, and Overall scores. We hypothesize that this advantage may stem, in part, from the quality of the synthetic editing supervision. In particular, the target images in the training dataset are generated by existing generative models rather than collected from real data, and may therefore inherit generation artifacts and model-specific biases. Directly supervising the model with such synthetic targets may propagate these imperfections, whereas **ControlEdit** avoids directly learning from the generated target images. Compared with NPEdit, another image editing method that does not rely on paired edited-image supervision, **ControlEdit** achieves substantially higher semantic consistency and Overall scores despite using a smaller base model. Moreover, **ControlEdit-L** consistently improves upon **ControlEdit-M**, indicating that our framework benefits from scaling to a larger generative backbone. Finally, compared with strong supervised image editing models (Liu et al., 2025; Wu et al., 2025; Labs et al., 2025), **ControlEdit** achieves competitive perceptual quality and strong instruction-following performance, despite not using paired edited images as direct training supervision. More details in Appendix A.1.

Qualitative analysis. We qualitatively compare **ControlEdit** with Qwen Image Edit (Wu et al., 2025), Step1x Edit (Liu et al., 2025), NP Edit (Kumari et al., 2025), and our supervised counterpart in Figure 3. Across diverse editing tasks, **ControlEdit** effectively follows editing instructions while preserving source content, including object pose, spatial layout, and scene context. In contrast, supervised baselines occasionally over-edit the source image, introducing unintended changes to its structure or appearance. NP Edit exhibits less consistent editing strength, producing insufficient edits in the pixel-art and bell-replacement examples, while over-editing the dog in the material-editing example. Overall, **ControlEdit** achieves a favorable balance between instruction alignment and content preservation.

Reward across training steps. Figure 4 shows the evolution of the training reward throughout RL post-training. The reward improves steadily during training, indicating that the model progressively learns to produce edits that better satisfy the reward objective. Performance reaches its peak at approximately 30K iterations, after which the reward begins to slightly decline. We therefore use the 30K-step checkpoint for evaluation, as further training does not yield additional improvement and may lead to over-optimization of the reward objective.

5.3 Ablation Study

Conditioning scale. To assess the effect of ControlNet conditioning strength, we replace the adaptive gating network with a fixed scale s, while keeping the LoRA checkpoint obtained after RL post-training. As shown in Table 2, a smaller s allows greater deviation from the source image, im-

proving editing flexibility but reducing content preservation, whereas a larger s strengthens source-image preservation at the cost of editing strength. Our adaptive strategy effectively balances these competing objectives and achieves the best overall VIEScore.

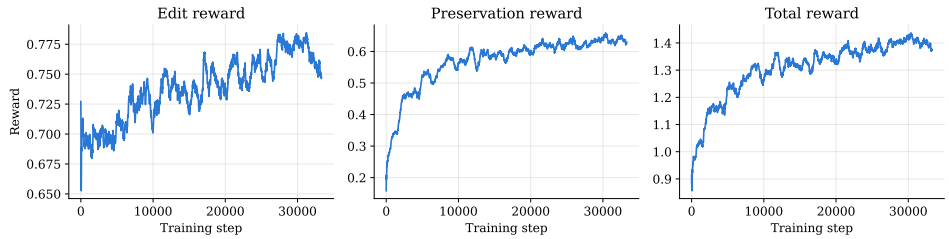

Figure 4: We visualize the reward across the training steps, including editing reward, preservation reward, and total reward. The reward steadily improves across training iterations.

Scale	Editing	Preservation	SC score	PQ score	Overall
0.6	0.86	0.36	5.24	6.28	5.09
0.7	0.71	0.44	5.41	6.68	5.21
0.8	0.60	0.58	5.91	6.95	5.86
0.9	0.45	0.72	5.63	6.89	5.49
Adaptive	0.78	0.58	**6.12**	**7.17**	**6.01**

Table 2: Importance of conditioning scale

Adaptive s	LoRA	SC Score	PQ Score	Overall
		5.76	6.67	5.42
✓		5.88	6.80	5.61
	✓	6.05	6.77	5.90
✓	✓	**6.12**	**7.17**	**6.01**

Table 3: Ablation study of different training components.

Depth size	SC score	PQ score	Overall
4	5.91	6.80	5.63
6	6.02	6.96	5.88
8	6.09	7.05	5.94
12	**6.12**	**7.17**	**6.01**

Table 4: Ablation study for the depth size of the ControlNet.

Ablation on training component. We conduct an ablation study to evaluate the contribution of each training component. Starting from the Stage-1 model, we independently examine the effects of adaptive ControlNet conditioning and LoRA adaptation of the main Transformer, with results reported in Table 3. When LoRA is disabled, the pretrained Transformer remains frozen and only the gating network is optimized during RL post-training; conversely, disabling adaptive conditioning removes the gating network while optimizing only the Transformer LoRA. Both components independently improve SC and PQ, while their combination achieves the best performance, demonstrating their complementary benefits for instruction following and perceptual quality.

Ablation on ControlNet depth. We further study the effect of ControlNet depth, which determines the capacity of the image-conditioning branch. As shown in Table 4, increasing the depth from 4 to our default 12 layers consistently improves the overall editing performance. This suggests that a deeper ControlNet provides a richer representation of the source image and enables more effective conditioning of the main Transformer. The results support our choice of a 12-layer ControlNet as the default configuration.

6 Conclusion

We present **ControlEdit** a two-stage framework that transforms pretrained text-to-image models into image editors without paired editing supervision. We first show that editing capability naturally emerges from ControlNet reconstruction pretraining, and further enhance this capability through RL post-training with adaptive ControlNet conditioning and Transformer LoRA. **ControlEdit** achieves competitive performance against supervised editing methods, demonstrating that strong image editing can be derived directly from pretrained generative priors without learning from paired edited images.

REFERENCES

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18392–18402. IEEE, 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 *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22503–22513. IEEE, 2023.

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis, 2023.

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

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.

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

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36: 52132–52152, 2023.

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

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

Inbar Huberman-Spiegelglas, Vladimir Kulikov, and Tomer Michaeli. An edit friendly ddpm noise space: Inversion and manipulations. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12469–12478. IEEE, 2024.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in neural information processing systems*, 36:36652–36663, 2023.

Max Ku, Dongfu Jiang, Cong Wei, Xiang Yue, and Wenhu Chen. Viescore: Towards explainable metrics for conditional image synthesis evaluation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12268–12290, 2024.

Nupur Kumari, Sheng-Yu Wang, Nanxuan Zhao, Yotam Nitzan, Yuheng Li, Krishna Kumar Singh, Richard Zhang, Eli Shechtman, Jun-Yan Zhu, and Xun Huang. Learning an image editing model without image editing pairs. *arXiv preprint arXiv:2510.14978*, 2025.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

WeiJie Li, Jin Wang, and Xuejie Zhang. Promptist: Automated prompt optimization for text-to-image synthesis. In *CCF international conference on natural language processing and Chinese computing*, pp. 295–306. Springer, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *Advances in neural information processing systems*, 38:40783–40818, 2026.

Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, et al. Step1x-edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11451–11461. IEEE, 2022.

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*, pp. 26627–26636, 2024.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6038–6047. IEEE, 2023.

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*, pp. 1–11, 2023.

Chau Pham, Quan Dao, Mahesh Bhosale, Yunjie Tian, Dimitris Metaxas, and David Doermann. Autoedit: Automatic hyperparameter tuning for image editing. *Advances in Neural Information Processing Systems*, 38:6174–6205, 2026.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations*, volume 2024, pp. 1862–1874, 2024.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22500–22510. IEEE, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

Qwen Team. Qwen3 technical report, 2025a.

Z-Image Team. Z-image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025b.

Yoad Tewel, Yuval Atzmon, Gal Chechik, and Lior Wolf. Bootstrap your generator: Unpaired visual editing with flow matching. *arXiv preprint arXiv:2606.03911*, 2026.

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

Bram Wallace, Akash Gokul, and Nikhil Naik. Edict: Exact diffusion inversion via coupled transformations. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22532–22541. IEEE, 2023.

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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8228–8238. IEEE, 2024.

Leyan Wang, Yonggang Jin, Tianhao Shen, Tianyu Zheng, Xinrun Du, Chenchen Zhang, Wenhao Huang, Jiaheng Liu, Shi Wang, Ge Zhang, et al. Giebench: Towards holistic evaluation of group identity-based empathy for large language models. *arXiv preprint arXiv:2406.14903*, 2024.

Yuhan Wang, Siwei Yang, Bingchen Zhao, Letian Zhang, Qing Liu, Yuyin Zhou, and Cihang Xie. Gpt-image-edit-1.5m: A million-scale, gpt-generated image dataset, 2025. URL https://arxiv.org/abs/2507.21033.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. URL https://arxiv.org/abs/2508.02324.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

Yang Ye, Xianyi He, Zongjian Li, Bin Lin, Shenghai Yuan, Zhiyuan Yan, Bohan Hou, and Li Yuan. Imgedit: A unified image editing dataset and benchmark. *arXiv preprint arXiv:2505.20275*, 2025.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3813–3824. IEEE, 2023.

Haozhe Zhao, Xiaojian 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.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. In *International Conference on Learning Representations*, volume 2026, pp. 134129–134150, 2026.

Method	Background	Color	Material	Motion	Human	Style	Add	Remove	Replace	Text	Tone	Mean
Qwen-Image-Edit	7.4	8.0	5.8	6.9	6.3	6.3	7.5	7.3	7.3	7.2	6.7	6.97
Step1x-Edit	7.6	8.2	7.5	6.9	6.7	7.5	7.8	7.2	8.0	8.8	7.5	7.61
Flux-Kontext	7.4	8.3	6.0	5.4	5.8	6.4	7.5	6.9	7.4	7.3	6.9	6.83
NP Edit	5.2	8.0	5.4	5.1	5.5	3.8	7.1	7.2	5.2	4.8	5.0	5.65
ControlEdit-M	6.6	7.0	6.5	6.6	6.0	6.8	7.0	3.7	6.4	4.7	5.8	6.12
ControlEdit-L	6.8	7.7	7.1	6.4	6.5	7.0	7.2	3.4	6.7	4.9	6.0	6.32

Table 5: SC Score of **ControlEdit** with other baselines for each editing task of GEdit-Bench dataset

Method	Background	Color	Material	Motion	Human	Style	Add	Remove	Replace	Text	Tone	Mean
Qwen-Image-Edit	7.6	7.2	7.2	7.8	7.7	7.1	7.7	7.6	7.7	7.4	7.3	7.49
Step1x-Edit	7.1	7.3	6.7	6.5	7.3	7.1	7.7	6.9	7.4	7.4	6.9	7.12
Flux-Kontext	7.3	6.9	7.0	7.4	7.1	6.7	7.5	7.3	7.4	7.2	6.9	7.15
NP Edit	7.5	7.4	7.2	7.7	7.5	7.5	7.5	7.2	7.2	7.2	7.3	7.40
ControlEdit-M	7.4	7.1	7.2	7.1	7.2	6.9	7.4	7.3	7.3	7.1	7.0	7.17
ControlEdit-L	7.4	7.2	7.3	7.3	7.4	7.3	7.6	7.5	7.6	7.2	7.3	7.38

Table 6: PQ Score of **ControlEdit** compared with other baselines for each editing task of the GEdit-Bench dataset

A Appendix

A.1 Additional quantitative results

We report the quantitative results for each editing tasks of the GEdit-Bench (Wang et al., 2024) dataset. There are a total of 11 editing tasks: Background change, Color change, Material Change, Motion Change, Human editing, Style transfer, Object-Add, Object-Remove, Object Replace, Text change, and Tone change. We report the SC Score in Table 5, PQ Score in Table 6 and Overall Score in Table 7.

Across all editing categories, the baseline supervised training model (Wu et al., 2025; Liu et al., 2025; Labs et al., 2025) achieves strong performance due to the strong training data and model architecture. Compared with another unsupervised method, NP Edit (Kumari et al., 2025), **ControlEdit** achieves better performance in Background change, Material Change, Motion change edit, Human editing, Style Transfer, and Object Replacement. For Object Removal, NP Edit achieves superior performance because its training strategy focuses on improving Object Removal.

We also compare **ControlEdit** with other methods in the ImgEdit benchmark (Ye et al., 2025). We report the Qwen2.5-VL-72B VieScore in Table 8. Also the results for each editing task are reported in Table 9, 10, 11.

A.2 Other choice of original image encoding

In this section, we consider an alternative design for encoding the source image based on token concatenation, following the conditioning strategy adopted in NP Edit (Kumari et al., 2025) and Bootstrap Generator (Tewel et al., 2026). Specifically, we encode the source image using the VAE and concatenate the resulting source-image latent tokens with the noisy latent tokens before feeding them into the main transformer. Full self-attention is then applied over the noisy latent tokens, text-

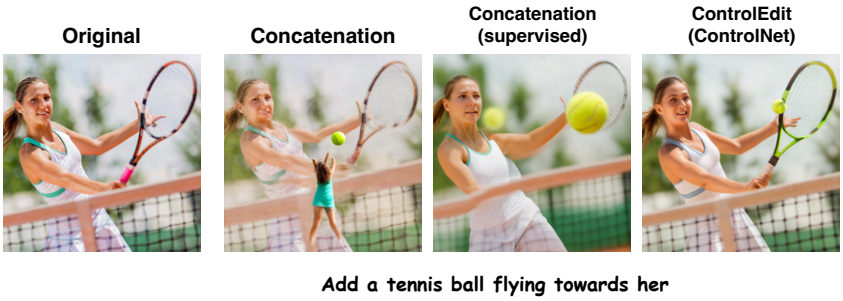

Figure 5: Visual illustrate of the comparison between ControlNet conditioning and concatenation conditioning.

Method	Background	Color	Material	Motion	Human	Style	Add	Remove	Replace	Text	Tone	Mean
Qwen-Image-Edit	7.27	7.36	5.58	6.83	6.45	6.11	7.34	7.02	7.12	6.50	6.59	6.97
Step1x-Edit	7.30	7.60	6.85	6.43	6.54	7.25	7.65	6.65	7.54	7.99	6.94	7.16
Flux-Kontext	7.21	7.44	5.67	5.38	5.62	6.07	7.31	6.39	6.84	6.71	6.75	6.49
NP Edit	5.31	7.55	5.19	5.15	5.50	3.61	6.85	6.54	5.13	4.52	5.10	5.50
ControlEdit-M	6.61	6.55	6.51	6.46	6.13	6.64	6.94	3.74	6.34	4.34	5.75	6.01
ControlEdit-L	6.86	7.01	6.84	6.89	6.34	6.84	7.01	3.48	6.49	4.83	5.96	6.23

Table 7: Overall score of **ControlEdit** with other baselines for each editing task of GEdit-Bench dataset

Method	**SC Score↑**	**PQ Score↑**	**Overall ↑**
Step1x Edit (Liu et al., 2025)	6.71	7.56	6.51
Qwen ImageEdit (Wu et al., 2025)	6.79	7.65	6.62
Flux-Kontext (Labs et al., 2025)	7.02	7.41	6.76
NP Edit (Kumari et al., 2025)	6.12	7.69	6.08
ControlEditM	6.25	7.46	6.14
Unpair Edit-L	6.37	7.57	6.28

Table 8: VIEScore on ImgEdit Bench

conditioning tokens, and source-image tokens, allowing the model to jointly reason over the source content and editing instruction. To adapt the pretrained transformer to image editing, we train LoRA modules using the same configuration as in our ControlNet-based variant, with a LoRA rank of 128. Since this formulation does not employ a separate ControlNet branch, we omit the gating network from the concatenation-based pipeline.

To enable subsequent RL post-training, we first perform a reconstruction warm-up stage analogous to Stage 1 of the ControlNet-based training. However, we observe that the concatenation-based design is more prone to overfitting to the source image and consequently ignoring the editing instruction. To mitigate this issue, we introduce several regularization strategies during reconstruction training, including:

Image conditioning dropout. To reduce the tendency of the model to over-rely on the source image, we drop the source-image conditioning for 60% of the training iterations. For these iterations, the model is trained as a standard text-to-image model using only the textual conditioning.

Regularization loss. We further introduce a directional regularization loss to discourage the model from trivially copying the source image:

$$\mathcal{L} = \|v_\theta(x_t, t, p_{\text{src}}, x_0^{\text{src}}) - v\|_2^2 + \lambda \left[1 - \cos(u_\theta(p_{\text{src}}, p_{\text{edit}}), u_{\theta_0}(p_{\text{src}}, p_{\text{edit}}))\right], \tag{13}$$

where

$$u_\theta(p_{\text{src}}, p_{\text{edit}}) = v_\theta(x_t, p_{\text{edit}}, x_0^{\text{src}}) - v_\theta(x_t, p_{\text{src}}, x_0^{\text{src}}), \tag{14}$$

$$u_{\theta_0}(p_{\text{src}}, p_{\text{edit}}) = v_{\theta_0}(x_t, p_{\text{edit}}) - v_{\theta_0}(x_t, p_{\text{src}}). \tag{15}$$

The first term is the standard reconstruction objective, while the second encourages the editing direction of the adapted model to remain aligned with the semantic direction induced by the pretrained text-to-image model. We set the regularization weight to $\lambda = 0.1$. Here, θ denotes the trainable LoRA parameters, while θ_0 denotes the frozen pretrained transformer.

We conduct RL post-training after the reconstruction stage and report the results in Table 12. The concatenation-based variant consistently underperforms the ControlNet-based design in the quantitative evaluation. We attribute this degradation to a higher tendency of the concatenation formulation to introduce hallucinated content during editing. In the ControlNet-based design, source-content preservation and instruction-driven modification are handled by separate modules, whereas in the concatenation variant, the same LoRA-adapted transformer must simultaneously preserve the source content and follow the editing instruction. This entanglement makes the model more prone to altering unrelated regions or introducing spurious content, which is reflected in the lower PQ score.

We further visualize qualitative comparisons in Figure 3. As shown in the figure, both the concatenation-based **ControlEdit** variant and the supervised concatenation baseline introduce hallucinated ball instances, while the ControlNet-based **ControlEdit** produces a single ball that better

Method	Action	Add	Adjust	Background	Compose	Extract	Remove	Replace	Style	Mean
Qwen-Image-Edit	7.25	7.94	7.03	7.52	5.13	4.77	6.44	8.23	6.82	6.79
Step1x-Edit	2.94	7.88	8.12	7.65	5.69	4.06	7.87	8.33	7.88	6.71
Flux-Kontext	6.67	7.83	7.59	7.85	7.04	4.87	6.31	8.18	6.85	7.02
NP Edit	6.32	7.56	7.80	6.97	5.04	2.83	7.21	6.72	4.58	6.12
ControlEdit-M	4.27	7.71	7.77	7.31	5.76	5.54	3.62	6.57	7.61	6.25
ControlEdit-L	4.48	7.82	7.85	7.45	5.95	5.69	3.75	6.65	7.72	6.37

Table 9: SC Score of **ControlEdit** compared with other baselines in ImgEdit Bench for each editing task

Method	Action	Add	Adjust	Background	Compose	Extract	Remove	Replace	Style	Mean
Qwen-Image-Edit	7.88	7.94	7.66	7.96	7.87	7.80	7.61	7.80	6.36	7.65
Step1x-Edit	7.86	7.94	7.66	7.73	7.82	7.76	6.83	7.30	7.12	7.56
Flux-Kontext	6.75	7.75	7.61	7.75	7.52	7.77	7.40	7.57	6.55	7.41
NP Edit	7.91	7.89	7.72	7.81	7.91	7.90	7.44	7.43	7.20	7.69
ControlEdit-M	7.74	7.75	7.41	7.44	7.44	7.86	7.63	7.32	6.56	7.46
ControlEdit-L	7.80	7.86	7.67	7.54	7.69	7.86	7.65	7.31	6.75	7.57

Table 10: PQ Score of **ControlEdit** compared with other baselines in ImgEdit Bench for each editing task

follows the editing instruction. We note that NP Edit mitigates this issue through its DMD-based training objective.

Although the concatenation-based variant underperforms our ControlNet formulation with the current training objective, this comparison also highlights the robustness of the proposed ControlNet-based decomposition. We hypothesize that concatenation-based conditioning could benefit from more specialized regularization or distillation objectives, and leave the development of such training strategies to future work.

A.3 Reward function

Following NP Edit (Kumari et al., 2025), we use a reward function that consists of editing alignment and content preservation by running a forward pass of the original image and edited image through the pretrained Qwen3-VL 4B model (Team, 2025a). We provide the prompt template below:

```
You are a professional digital artist and an expert image editor.  You
will be provided with two images.
The first being the original real image, and the second being an
edited version of the first image.
Editing instruction:  {editing instruction}
Rate how clearly and completely this instruction has been executed in
the second image, on a scale of 1 to 5:
1 = not executed at all (no visible sign of the change)
2 = barely executed (a faint, washed-out, or ambiguous trace only)
3 = partially executed (visible but weak, incomplete, or superficial)
4 = mostly executed (clearly visible, minor imperfections)
5 = fully executed (clear, vivid, unambiguous, complete, present
as a single correctly-placed and correctly-scaled instance -- not
duplicated, floating, or disconnected from its expected context)
If the edit introduces a duplicated, floating, disconnected, or
badly misplaced/mis-scaled version of the element (even if a correct
instance is also present elsewhere in the image), that counts against
the score -- do not rate a duplicated or badly misplaced result higher
than a 3.
Answer with only a single digit:  1, 2, 3, 4, or 5.
```

```
You are a professional digital artist and an expert image editor.  You
will be provided with two images.
```

Method	Action	Add	Adjust	Background	Compose	Extract	Remove	Replace	Style	Mean
Qwen-Image-Edit	7.22	7.94	6.67	7.51	5.41	4.83	6.02	7.82	6.20	6.62
Step1x-Edit	2.99	7.86	7.73	6.53	6.15	4.05	7.09	7.71	7.44	6.51
Flux-Kontext	6.33	7.73	7.19	7.78	7.05	4.91	5.81	7.47	6.32	6.76
NP Edit	6.28	7.45	7.70	7.05	5.53	2.94	6.94	6.38	4.45	6.08
ControlEdit-M	4.26	7.58	7.53	7.15	6.01	5.74	3.55	6.33	6.97	6.14
ControlEdit-L	4.37	7.72	7.67	7.40	6.15	6.11	3.60	6.52	7.06	6.28

Table 11: Overall score of **ControlEdit** with other baselines in ImgEdit Bench for each editing task

Mechanism	SC Score	PQ Score	Overall
ControlNet	6.12	**7.17**	6.01
Concatenation	5.94	6.48	5.71
Concatenation (Supervised)	**6.23**	6.88	**6.03**

Table 12: Ablation study of different conditioning modules

```
The first is the original image and the second is an edited version.
The editing instruction was:  "{edit instruction}"
IGNORE the part that was edited according to the instruction.  Rate
how well everything else (objects, background, colors, layout,
lighting) is preserved in the second image, on a scale of 1 to 5:
1 = not preserved at all, unrecognizable
2 = poorly preserved, major unintended changes
3 = partially preserved, some noticeable unintended changes
4 = mostly preserved, only minor unintended changes
5 = fully preserved, no unintended changes
Answer with only a single digit:  1, 2, 3, 4, or 5.
```

And the score computation is given by:

$$s = \frac{1}{4}(\sum_{i=1}^{5} i * \text{softmax}(\{l_j\}_{j=1}^{5})_i - 1) \tag{16}$$

where l_i is the logit of character i in the forward pass of the QwenVL model.

A.4 Training details

In this section, we provide the detailed implementation of the Gating network and the training details of **Unpair**.

Gating network implementation We use the layer-wise ControlNet features, text embeddings, and diffusion timestep t as inputs to the gating network G_ϕ. The network contains K layer-specific modules, where K denotes the number of ControlNet blocks. For each layer, the ControlNet and text features are first projected into a shared feature space and fused through an attention module. The resulting features are spatially pooled and concatenated with the timestep embedding obtained from a time-projection layer. Finally, a two-layer MLP with SiLU activation predicts a scalar conditioning scale for each ControlNet block. A sigmoid function constrains the predicted scale to the range $(0, 1)$.

Stage-1 training. We use the 12-block ControlNet implementation from Diffusers (von Platen et al., 2022). The ControlNet is initialized from the pretrained Transformer weights and optimized for source-image reconstruction with the conditioning scale fixed to $s = 1$. We train for 2K iterations on 4 GPUs with a per-GPU batch size of 8.

Warm-up training. We introduce LoRA modules to the main Transformer using the configuration of DiffusionNFT (Zheng et al., 2026), with a LoRA rank of 128. We optimize the instruction-distillation objective in Equation 5 while uniformly sampling the conditioning scale from $[0.4, 1.0]$. The ControlNet is frozen during this stage, and only the LoRA parameters are trained for 3K iterations on 4 GPUs with a per-GPU batch size of 8.

RL post-training. In the final stage, we jointly optimize the Transformer LoRA parameters and the gating network G_ϕ to maximize the editing reward. The LoRA parameters are initialized from the warm-up stage. For each source image-instruction pair, we use a rollout group size of $G = 16$, generating 16 candidate edited images. Qwen3-VL-4B evaluates each candidate according to our editing criteria, and the advantage is standardized using the mean and standard deviation of rewards within the same rollout group. Following DiffusionNFT (Zheng et al., 2026), we further normalize and clip the advantage to $[0, 1]$:

$$a \leftarrow \mathrm{Clip}\left(0.5\frac{a}{a_{\max}} + 0.5,\ 0,\ 1\right), \qquad a_{\max} = 5. \tag{17}$$

For each training iteration, we roll out 4 source image–instruction pairs per GPU, with 16 candidates per pair, resulting in 64 generated images per GPU. We use a regularization coefficient of $\lambda = 0.01$ and an exponential moving-average coefficient of $\gamma = 0.5$ for the DiffusionNFT update.

A.5 Training algorithm

We provide the training details in Algorithm 1.

A.6 Additional qualitative results

We provide the additional qualitative results in Figure 6 and 7 for **ControlEdit** and different baselines.

A.7 Limitation

One limitation of **UnpairEdit** is its performance on removal editing. As shown in Tables 5 and 9, our method underperforms NP Edit and other baselines on this category. We observe that the Stage-1 model already struggles to remove target objects, and the subsequent RL post-training provides only limited improvement. We therefore leave robust object removal as an important direction for future work.

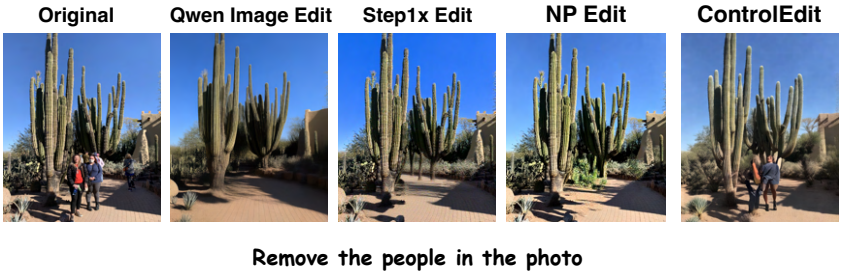

Figure 8: We visualize the limitation of Object Removal task, where Unpair Edit struggles to remove the object in some scenario

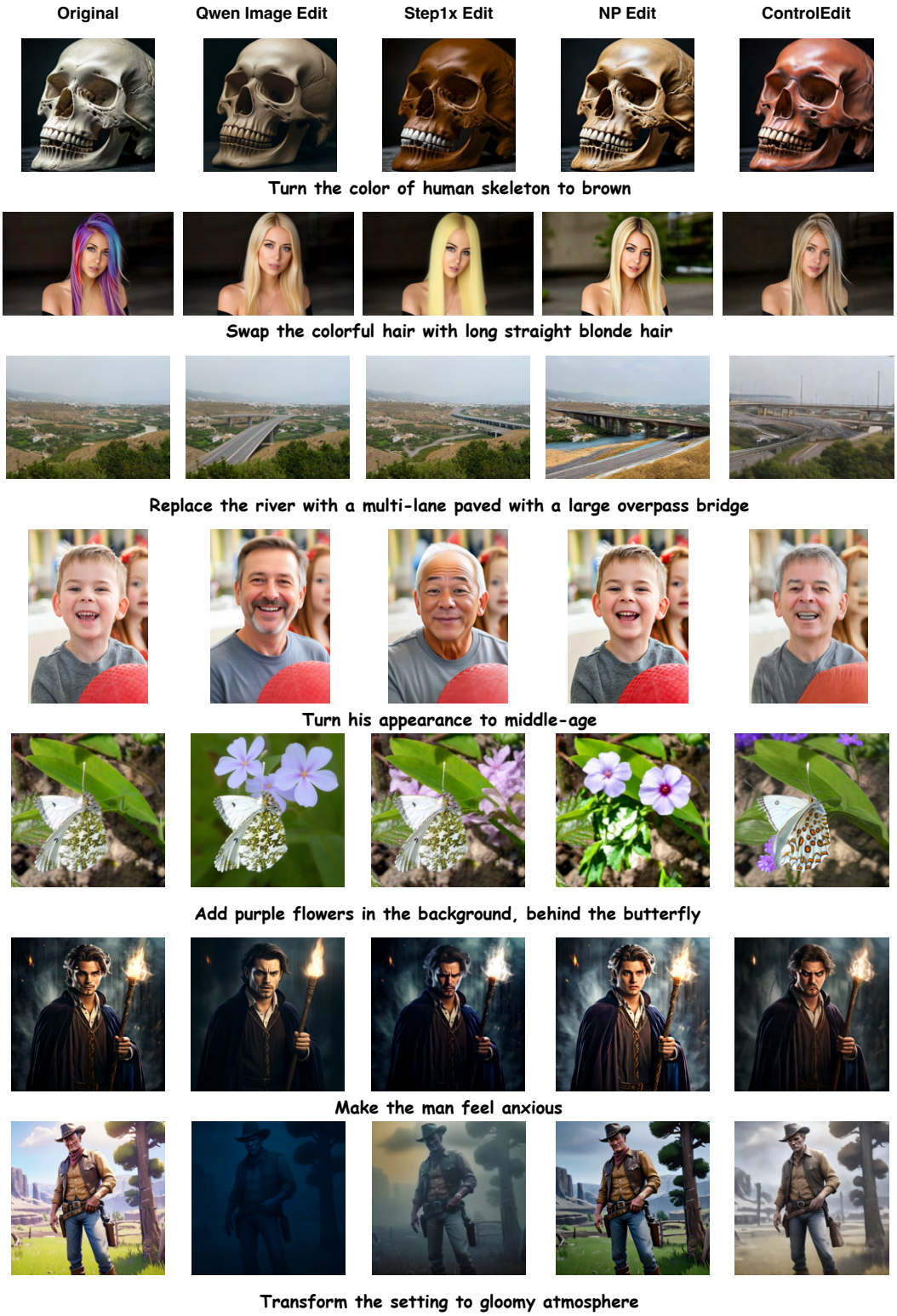

Figure 6: Additional qualitative examples of **ControlEdit** and other baselines

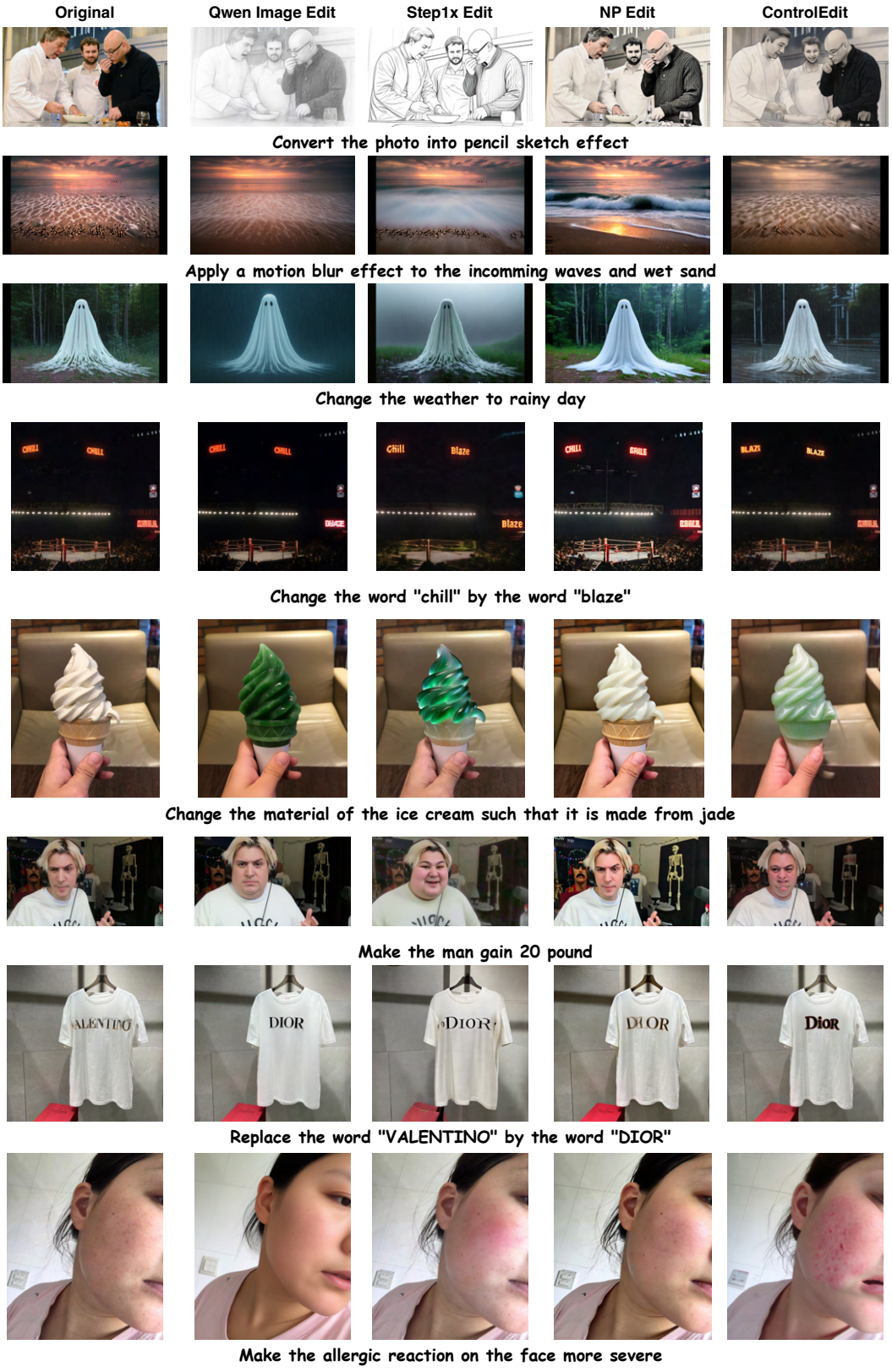

Figure 7: Additional qualitative examples of **ControlEdit** and other baselines

Algorithm 1: ControlEdit training mechanism

Input: Pretrained text-to-image model v_θ, VLM reward model R, dataset $\{(I^{\text{src}}, p_{\text{src}}, p_{\text{edit}}, p_{\text{instr}})\}$
Output: Image-to-image editing model v_θ^{edit}

// Initialize the ControlNet
$C_\tau \leftarrow \text{Initweight}(v_\theta)$
for step $= 1$ **to** N_{stage1} **do**
  $\epsilon \sim \mathcal{N}(0, I), t \sim U(0,1), v_t = (1-t)x_0^{src} + t\epsilon$
  $\mathcal{L}_{\text{stage1}} = \|v_\theta(x_t, t, p_{\text{src}}, C_\tau(x_t) - v\|^2, v = \epsilon - x_0^{src}$
  $\tau = \tau - \eta\nabla\mathcal{L}_{\text{stage1}}$
// A small distillation to initialize the model with instruction prompt
$v_{\theta'} \leftarrow \text{LoRAInit}(v_\theta)$
for step $= 1$ **to** N_{stage1} **do**
  $\epsilon \sim \mathcal{N}(0, I), t \sim U(0,1), v_t = (1-t)x_0^{src} + t\epsilon, c_\tau = C_\tau(x_0^{src})$
  $v_{\text{instr}} = v_{\theta'}(x_t, t, p_{\text{instr}}, c_\tau)$, $v_{\text{edit}} = v_\theta(x_t, t, p_{\text{edit}}, c_\tau)$, $v_{\text{src}} = v_\theta(x_t, t, p_{\text{src},c_\tau})$
  $\mathcal{L}_{\text{distill}} = 1 - \cos(v_{instr} - v_{src}, v_{edit} - v_{src}) + \lambda\|v_{instr} - v_{edit}\|^2$
// Final RL post-training
// Initialize the components of RL
$\theta_{\text{old}} \leftarrow \theta', \theta \leftarrow \theta'$
$\phi \leftarrow \text{Init}(\phi)$
$\phi_{\text{old}} \leftarrow \phi$
for step $= 1$ **to** N_{RL} **do**
  // Rollout phase: Given the original image x_0^{src} and p_{instr}, we need to generate G different images of x_0^{src} and p_{instr}
  for $t = T$ **to** *0* **do**
    $c\tau = C_\tau(x_t, t)$
    $s_1, s_2, ..., s_K = G_{\phi_{\text{old}}}(c_\tau, p_{\text{instr}}, t)$
    $v_{\text{pred}} = v_{\theta_{\text{old}}}(x_t, t, C_\tau, \{s_i\})$
    $v_{t-1} = \text{Step}(x_t, v_{pred}, t)$
  Grab $x_0^1, x_0^2, ..., x_0^G$ as G x_0 sampled for $x^{\text{src}}, p_{\text{instr}}$ with G be the group size.
  // Reward and advantage computation
  $r_i = R(x^i, x_0, p_{\text{instr}})$
  $a_i = \frac{r_i - \text{mean}(\{r_i\}_{i=1}^G)}{\text{std}(\{r_i\}_{i=1}^G)}$
  // Advantage normalization and relative computing
  $r^i = \text{Clip}(\frac{1}{2}\frac{a^i}{a_{\max}} + \frac{1}{2}, 0, 1)$
  // NFT training
  for $t = 1$ **to** T **do**
    for $i = 1$ **to** G **do**
      $x_t^i = (1-t)x_0^i + t\epsilon$
      $S_{\text{old}} = G_{\phi_{old}}(c_\tau, t, p_{\text{instr}})$
      $S_{\text{fwd}} = G_\phi(c_\tau, t, p_{\text{instr}})$
      $v_{\text{old}} = v_{\theta_{\text{old}}}(x_t, t, p_{\text{instr}}, c_\tau, s = S_{\text{old}})$
      $v_{\text{fwd}} = v_\theta(x_t, t, p_{\text{instr}}, c_\tau, s = S_{\text{fwd}})$
      $v_{\text{ref}} = v_{\theta'}(x_t, t, p_{\text{instr}}, s = 1)$
      $v^+ = \beta v_{\text{fwd}} + (1-\beta)v_{\text{old}}$
      $v^- = (1-\beta)v_{\text{fwd}} + \beta v_{\text{old}}$
      $\mathcal{L}_{policy} = r^i\|v^+ - v\|^2 + (1-r^i)\|v^- - v\|^2$ with $v = \epsilon - x_0^i$
      $\mathcal{L}_{reg} = \|v_{ref} - v_{fwd}\|^2$
      $\mathcal{L}_{final} = \mathcal{L}_{policy} + \lambda\mathcal{L}_{reg}$
      $\phi, \theta = \phi, \theta - \eta\nabla\mathcal{L}_{final}$
      Update old network: $\theta_{\text{old}} = \gamma\theta_{\text{old}}\theta_{\text{old}} + (1-\gamma)\theta$
      $\phi_{\text{old}} = \gamma\phi_{\text{old}} + (1-\gamma)\phi$