AnyEdit: Mastering Unified High-Quality Image Editing for Any Idea

Supplementary Material

A. Overview

In this supplementary material, we present:

- More detailed dataset collection process of AnyEdit (Section B).
- Statistic information of AnyEdit (Section C).
- Additional examples of AnyEdit (Section D).
- Detailed description of AnyEdit-Test Benchmark (Section E).
- Detailed experimental results of various editing types on AnyEdit-Test Benchmark (Section F).
- Implementation details (Section G).
- More qualitative results on various benchmarks and human evaluations (Section H).

B. Detailed Dataset Collection Process

B.1. Editing Type Definition

Here, we explain the detailed definition of each editing task in AnyEdit, which comprises five primary categories with 25 distinct editing types, as shown in Table 7.

B.2. Diverse Instruction Generation.

B.2.1. Prompt Constraints

To address the limitations in instruction diversity and consistency during the process of Instruction Generation, we use prompt constraints to guide the LLM as a task-specific agent that responds in JSON format with diverse editing types. A key innovation is incorporating a task-specific user-LLM conversational history into the prompt, where we replace direct constraints with high-quality, handcrafted examples. This approach enables the LLM to learn from ideal responses and improve subsequent generations. Specifically, we design templates for each task instructing the LLM to respond in the required format. These templates are tailored for each task and include four core elements: *input*, *output*, *editing type*, and *edit instruction*. We also define a set of action verbs for each task, ensuring the LLM's response aligns with our guidelines and improving the quality and consistency of the generated output. The detailed prompt constraints are shown in Table 8.

B.2.2. In-context Examples

As mentioned in Section 3.2, we employ in-context examples in the conversation history to develop a task-specific agent tailored to each editing type. For each new task, we initially generate a set of five in-context example instructions into prompt templates for instruction generation. After generating an editing instruction, we integrate it with its original caption to create instruction pairs, which are used to expand the in-context example pool. For each subsequent generation process, five in-context examples are randomly selected. This iterative self-enhancement mechanism exposes the generation process to a diverse range of examples, encouraging more varied and robust output responses. In this way, we maintain a cohesive conversational flow while progressively increasing the diversity and complexity of the generated instructions, facilitating the construction of the AnyEdit dataset.

B.3. Adaptive Editing Pipelines

This section will elaborate on the pipeline implementation details for various editing tasks in the AnyEdit datasets collection. Figure 1 shows the illustrations of the main specific pipelines in our Adaptive Editing Pipeline module to construct these various high-quality editing instructions adaptively.

B.3.1. Local Editing

Remove. As shown in Pipeline1 of Figure 1, we first extract the mask of the edited object in the editing instruction by GroundingDINO [16] and Segment Anything [13]. We then generate the target image using SD-Inpaint [29], with the original image and the mask produced during the process. To remove the target object, we set the prompt word to empty and the negative prompt word to edited object. Notably, we also apply dilation to the mask and perform Gaussian filtering to smooth it, ensuring a more natural blend with the surrounding contents. In addition, we merge the edited image, mask, and original image to retain image elements outside the content of the editing instructions.

Replace. As shown in Pipeline1 of Figure 1, the process of generating data for replace type is similar to the removal. The only difference is that we set the prompt word to the new object. In this way, our pipeline tends to produce a new object to replace the edited object instead of removing it.

Add. As shown in Pipeline1 of Figure 1, the process of generating data for add type is similar to the removal. However, since the add edit is to add a new object to the original image, its correct placement is unknown. Thus, we reverse the process by first generating the image of the output caption as the edited image and then using the 'remove' editing instruction to obtain the original image that does not include the newly added object. In this way, we obtain the original image and the edited image with the newly added object seamlessly integrated.

Counting. The counting-type editing introduces the concept of object quantity, performing the corresponding num-

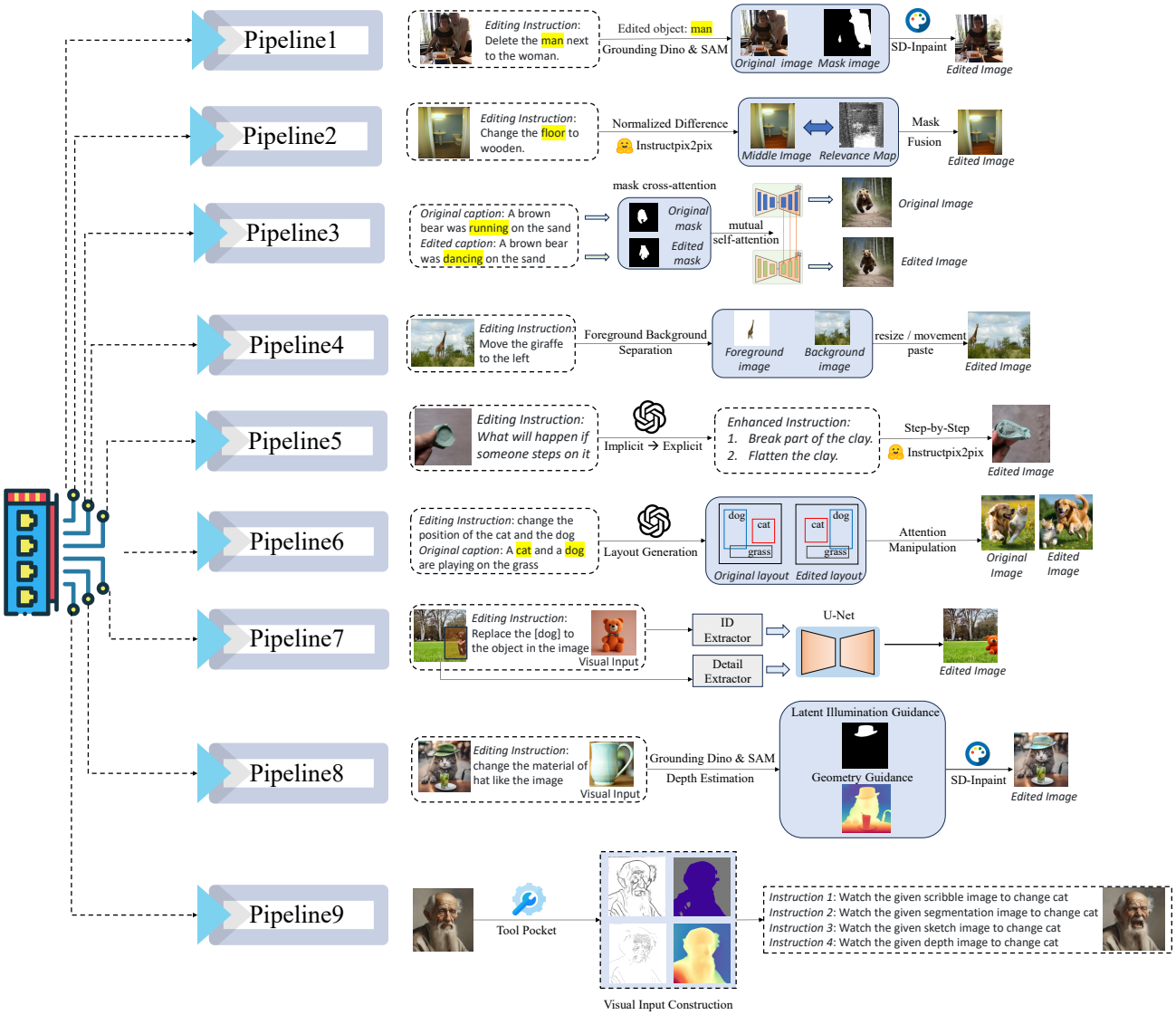


Figure 1. The illustration of detailed pipelines for each editing type in the AnyEdit dataset collection.

ber of removal or additions iteratively based on the specified count in the instructions. Each step in the process is illustrated in the Pipeline1 of Figure 1.

Color Alter & Appearance Alter. As shown in Pipeline2 of Figure 1, the key aspect of editing types like color alteration and appearance alteration lies in modifying only the object's attributes or appearance instead of altering or removing whole objects. Therefore, we introduce a Normalized Attention Difference [19] based on input-output caption discrepancies to identify the target editing mask. Based on this, we apply InstructPix2Pix [2] for instruction-based editing, blending the original and edited images within the masked region to produce the final result, thereby minimizing element confusion.

Action Change. To achieve complex non-rigid image editing, we introduce a joint intervention mechanism involving mutual self-attention and masked cross-attention, as shown in Pipeline3 of Figure 1. This approach addresses the limitations in the action change editing instructions, which sometimes fail to accurately execute editing intentions due to the need for fine-grained modifications.

Textual Change. To meet the demands for textual change, we collected captions containing text from the AnyWord-3M dataset [35], namely, ArT [5], COCO-Text [6], RCTW [27], LSVT [17], MLT [20], MTWI [23], ReCTS [28]. Following this, we generate editing instructions that alter only the text within the image, guided by specific type constraints and in-context examples in our diverse instruction generation. We ultimately generate corresponding images as the final result by using a text-specialized T2I

model (*i.e.*, FLUX), with both the original caption and the edited caption maintained under the same seed.

Material Change. We reuse the original and edited images from Material Transfer in Visual Editing. However, we only utilize editing instructions to convey the editing intent without using material images as references. Specifically, we will change "the material of [object] like the image" to "change the material of [object] to [material category]".

B.3.2. Global Editing

Background Change. As shown in Pipeline1 of Figure 1, we define background changes as modifications to the edited object "background". To avoid unnecessary foreground modifications, we extract and invert all foreground masks from captions, then merge them with the background mask. Similar to the replacement instructions, we also apply dilation to the merged mask and perform Gaussian filtering to eliminate artifacts in the contour.

Tone Transfer. We define three types of changing scenes (*i.e.*, season, time, weather) involving the overall tone of the image. According to this, we generated editing instructions tailored to tone transfer and used InstructPix2Pix [2] to edit the whole image, as shown in Pipeline1 of Figure 1.

Style Change. We collect 50 desired style images and extract 2,500 images from the MSCOCO validation set as original images. Using an API of Prisma Art, we applied style transfer to obtain the edited results. Ultimately, we only retain the intuitive style as the style changes editing instructions, such as "animated".

B.3.3. Camera Movement Editing

Movement & Resize. As shown in Pipeline4 of Figure 1, we first extract the foreground object and backgrounds separately according to the edited object. Here we use the "remove" operation to ensure the pixel integrity of the background after removing the foreground. Then, we utilize the crop-and-paste operation to change the size of the edited object and the position of it in the edited image.

Outpainting. To reduce the complexity of constructing data, we inversely designate the images from the initial dataset as extended images. Given the input caption, we randomly select an object within it and use GroundingDINO to extract its bounding box in the image. Then, we apply a mask to the areas outside the bounding box and obtain the original image that contains only selected elements. The extended and original images are then used to construct editing instructions for the outpainting type.

Rotation Change. Since direct perspective rotation change of images is challenging, we extract related image pairs directly from MVImgNet [40], the Large-scale Dataset of Multi-view Images, to construct original images and edited images for rotation changes. Then, we categorize the editing instructions as "rotate the object clockwise" and "rotate the object counterclockwise" according to changes in the camera's viewpoint, thereby constructing corresponding pairs of editing data.

B.3.4. Implicit Editing

Implicit Change. As shown in Pipeline5 of Figure 1, we first elicit the world knowledge from LLMs to transform implicit instructions into explicit instructions, which directly convey executable editing intentions (*e.g.*, "Flatten the clay" directly conveys the alteration in the clay's appearance without requiring additional interpretation). In this way, we use the instruction-based editing method to complete the explicit instructions step-by-step, thereby constructing edited images with implicit changes. We also enrich the dataset by using existing dynamic world editing datasets [38].

Relation Change. To adjust the positional relationships of objects within images, our pipeline first generates layouts based on the original captions, as shown in Pipeline6 of Figure 1. We can swap the positional relationship between two objects in the layout space to construct the edited layout. Subsequently, we adopt attention manipulation to the layout-to-image models [46] to generate original and edited images that alter only relative positioning without changing other content.

B.3.5. Visual Editing

Image Reference. We are the first to incorporate additional visual input into instruction-based image editing. To reduce the cost of automated synthetic data generation, we leverage zero-shot image customization [4] to synthesize images containing visual concepts. We repurpose the edited objects from the remove or replace steps and the corresponding masks to guide the target positioning within the edited image. Additionally, we introduce an ID extractor to embed visual concepts into the target image and a detail extractor to preserve fine content details. Finally, We construct edited images containing the visual concepts in the image reference, as shown in the pipeline7 of Figure 1.

Material Transfer. Similar to the image reference, the material transfer requires injecting the material reference into the target image to achieve the material transfer effect. Considering the compatibility between materials and target objects, we further introduce depth estimation and latent illumination guidance for seamless material fusion. The total process is shown in the Pipeline8 of the Figure 1.

Visual Condition. To support a broader range of visual editing types, we incorporate additional condition images as reference images through tool pockets from ControlNet [42]. We use tools to generate the corresponding conditional images and construct the corresponding editing instructions by templates. Notably, the edited images originate from other editing instructions, where the visual condition constructs new instruction pairs without generating

additional edited images.

C. Statistics of AnyEdit

We present detailed dataset statistics for all editing types in AnyEdit in Table 1.

Editing Type	#Instruction	#Image
Local Editing		
Remove	116013	116013
Replace	97219	97219
Add	390049	390049
Color Alter	337078	337078
Appearance Alter	79720	79720
Material Change	21646	-
Action Change	47210	47210
Textual Change	2500	2500
Counting	698	698
Global Editing		
Background Change	413570	413570
Tone Transfer	553919	553919
Style Change	27488	-
Camera Movement Editing		
Movement	7724	7724
Outpaint	57462	57462
Rotation Change	17022	17022
Resize	10219	10219
Implicit Editing		
Implicit Change	10000	10000
Relation Change	410	410
Visual Editing		
Visual Sketch	55385	-
Visual Scribble	55385	-
Visual Segmentation	55385	-
Visual Depth	55385	-
Visual Layout	55385	-
Material Transfer	21646	21646
Image Reference	17885	17885
Total	2506403	2180350

Table 1. The detailed statistics of the AnyEdit dataset.

D. More Examples of AnyEdit

See figure 2 and 3 for more data examples with various editing types in AnyEdit.

E. AnyEdit-Test Benchmark

To comprehensively evaluate AnyEdit's capabilities across a broader range of editing tasks, we carefully selected 50 example pairs from each type of editing task supported by AnyEdit. This selection process allowed us to construct a new test set, named **AnyEdit-Test**, designed specifically to provide a more rigorous assessment. The resulting dataset includes diverse and representative editing challenges, offering a well-rounded evaluation benchmark better to understand AnyEdit's performance across different task types. In this way, AnyEdit-Test not only broadens the scope of evaluation but also ensures that the test set includes a variety of editing complexities, thereby making the evaluation both more challenging and more insightful. In figures 4 to 8, the editing examples encompass all types from our AnyEdit dataset, all of which demonstrate excellent adherence to instructions and high visual fidelity. This further attests to the high quality and diversity of the editing data in AnyEdit. We will release this high-quality dataset and benchmark for community research.

F. Detailed Experiments of AnyEdit-Test

We conduct detailed quantitative evaluations of 25 editing types in AnyEdit-Test, focusing on editing accuracy and content consistency. Detailed results for the AnyEdit-Test benchmark can be seen in Table 9, 10, 11. We have the following observations: (1) Existing models often fail to ensure editing accuracy in complex tasks (*e.g.*, significant reduction of CLIPim in action change, rotation, outpainting shown in Tab. 9 & 10, exposing the limitations of current benchmarks for complex tasks. For fine-grained editing tasks, models frequently struggle to maintain the integrity of image content while making precise modifications (*e.g.*, L1 nearly doubled performance degradation in action change and textual change). These tasks demand both a high level of fine-grained control and the ability to preserve the original context. These limitations highlight a fundamental gap in existing benchmarks and AnyEdit-Test, which is more comprehensive for complex, real-world editing demands. (2) Even for common tasks in AnyEdit-Test, some previous SOTA models show a notable performance drop compared to existing benchmarks, revealing the limitations of current benchmarks in multi-scene editing. While many state-of-the-art models have achieved impressive results on conventional benchmarks, they struggle to generalize to more diverse, multi-scene editing tasks that are present in AnyEdit-Test. This performance drop highlights the limitations of traditional benchmarks when adapting to the increased diversity of multi-scene editing. In contrast, AnyEdit-Test introduces a broader range of editing scenes, making it a more accurate reflection of real-world scenarios.

G. Implementation Details

G.1. AnySD Architecture

AnySD is a diffusion model designed to handle a broad range of editing tasks through language-based instructions. Given the distinct demands of each edit type, which require the model to selectively focus on different elements—such

as faithfully preserving visual likeness in visual instructions or altering style while retaining the original image composition in style transfer—we adopt a Mixture of Experts (MoE) architecture [18].

The visual condition c_V is integrated into the pretrained UNet [31] by the adapted modules with decoupled cross-attention to avoid disrupt the edit instruction condition. Each MoE block share the same language attention layer but diverse in the attention for c_V and the weights are distributed by the router based on the task embedding.

In the original SD model, given the query features $\mathbf{Z}$ and the text features $\boldsymbol{z}_t$, the output of cross-attention $\mathbf{Z}'$ can be defined by the following equation:

$$\mathbf{Z}' = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V}, \quad (1)$$

where $\mathbf{Q} = \mathbf{Z}\mathbf{W}_q$, $\mathbf{K} = \boldsymbol{z}_t\mathbf{W}_k$, $\mathbf{V} = \boldsymbol{z}_t\mathbf{W}_v$ are the query, key, and values matrices of the attention operation respectively, and $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$ are the weight matrices of the trainable linear projection layers.

To achieve separate attention mechinism, we add a new cross-attention layer for each cross-attention layer in the original UNet model to insert image features. Given the $\boldsymbol{c}_V$, the output of new cross-attention $\mathbf{Z}''$ is computed as follows:

$$\mathbf{Z}'' = \text{Attention}(\mathbf{Q}, \mathbf{K}', \mathbf{V}') = \text{Softmax}(\frac{\mathbf{Q}(\mathbf{K}')^\top}{\sqrt{d}})\mathbf{V}', \quad (2)$$

where, $\mathbf{K}' = \boldsymbol{c}_v\mathbf{W}'_k$ and $\mathbf{V}' = \boldsymbol{c}_v\mathbf{W}'_v$ are the query, key, and values matrices from the image features. $\mathbf{W}'_k$ and $\mathbf{W}'_v$ are the corresponding weight matrices. In order to speed up the convergence, $\mathbf{W}'_k$ and $\mathbf{W}'_v$ are initialized from $\mathbf{W}_k$ and $\mathbf{W}_v$. Then, we simply add the output of image cross-attention to the output of text cross-attention:

$$\mathbf{Z}^{new} = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V} + \text{Softmax}\left(\frac{\mathbf{Q}(\mathbf{K}')^\top}{\sqrt{d}}\right)\mathbf{V}' \quad (3)$$

G.2. CFG for Three Conditionings

AnySD is based on the latent diffusion model architecture [29, 30, 34] to support high-resolution image generation and incorporated variational autoencoder [12] with encoder $\mathcal{E}$ and decoder $\mathcal{D}$, with estimating the score [9] of a data distribution. To support image conditioning, we add additional input channels to the first convolutional layer on the simple text to image diffusion model [30], concatenating z_t and $\mathcal{E}(c_I)$, following InstructPix2Pix [2].

For an image x, the diffusion process adds noise to the encoded latent $z = \mathcal{E}(x)$ producing a noisy latent z_t where the noise level increases over timesteps $t \in T$. We learn a network ϵ_θ that predicts the noise added to the noisy latent z_t given original image conditioning c_I, text edit instruction conditioning c_T and visual prompt conditioning c_V. We minimize the following latent diffusion objective:

$$L = \mathbb{E}_{\mathcal{E}(x),\mathcal{E}(c_I),c_T,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t, \mathcal{E}(c_I), c_T, c_V))\|_2^2\right] \quad (4)$$

Classifier-free diffusion guidance (CFG) [7, 15] effectively shifts probability mass toward data where an implicit classifier $p_\theta(c|z_t)$ assigns high likelihood to the conditioning c. Training for unconditional denoising is done by simply setting the conditioning to a fixed null value $c = \varnothing$ at some frequency during training. At inference time, with a guidance scale $s \geq 1$, the modified score estimate $\tilde{e_\theta}(z_t, c)$ is extrapolated in the direction toward the conditional $e_\theta(z_t, c)$ and away from the unconditional $e_\theta(z_t, \varnothing)$.

$$\tilde{e_\theta}(z_t, c) = e_\theta(z_t, \varnothing) + s \cdot (e_\theta(z_t, c) - e_\theta(z_t, \varnothing)) \quad (5)$$

For our task, the score network $e_\theta(z_t, c_I, c_T, c_V)$ has three conditionings: the input image c_I, text instruction c_T and visual prompt c_V. We introduce two guidance scales, s_I, s_T and s_V which can be adjusted to trade off how strongly each condition. Our modified score estimate is as follows:

$$\begin{aligned}\tilde{e_\theta}(z_t, c_I, c_T, c_V) = \; & e_\theta(z_t, \varnothing, \varnothing, \varnothing) \\ & + s_I \cdot (e_\theta(z_t, c_I, \varnothing, \varnothing) - e_\theta(z_t, \varnothing, \varnothing, \varnothing)) \\ & + s_T \cdot (e_\theta(z_t, c_I, c_T, \varnothing) - e_\theta(z_t, c_I, \varnothing, \varnothing)) \\ & + s_V \cdot (e_\theta(z_t, c_I, c_T, c_V) - e_\theta(z_t, c_I, c_T, \varnothing))\end{aligned} \quad (6)$$

G.3. Classifier-free Guidance Details

As discussed in Section G.2, we apply classifier-free guidance with respect to three conditionings: the input image c_I , the text instruction c_T and the visual prompt with task embedding c_V. We introduce separate guidance scales s_I, s_T and s_V that enable separately trading off the strength of each conditioning.

When ignoring c_V, we can have the modified score estimate as InstructPix2Pix [2]:

$$\begin{aligned}\tilde{e_\theta}(z_t, c_I, c_T) = \; & e_\theta(z_t, \varnothing, \varnothing) \\ & + s_I \cdot (e_\theta(z_t, c_I, \varnothing) - e_\theta(z_t, \varnothing, \varnothing)) \\ & + s_T \cdot (e_\theta(z_t, c_I, c_T) - e_\theta(z_t, c_I, \varnothing))\end{aligned}$$

Below is the modified score estimate for our model with classifier-free guidance on three conditions (copied from Equation 6):

$$\begin{aligned}\tilde{e_\theta}(z_t, c_I, c_T, c_V) = \; & e_\theta(z_t, \varnothing, \varnothing, \varnothing) \\ & + s_I \cdot (e_\theta(z_t, c_I, \varnothing, \varnothing) - e_\theta(z_t, \varnothing, \varnothing, \varnothing)) \\ & + s_T \cdot (e_\theta(z_t, c_I, c_T, \varnothing) - e_\theta(z_t, c_I, \varnothing, \varnothing)) \\ & + s_V \cdot (e_\theta(z_t, c_I, c_T, c_V) - e_\theta(z_t, c_I, c_T, \varnothing))\end{aligned}$$

Our generative model learns $P(z|c_I, c_T)$, the probability distribution of image latents $z = \mathcal{E}(x)$ conditioned on an input image c_I, a text instruction c_T and the visual prompt with task embedding c_V. We arrive at our particular classifier-free guidance formulation by expressing the conditional probability as follows:

$$P(z|c_T, c_I, c_V) = \frac{P(z, c_T, c_I, c_V)}{P(c_T, c_I, c_V)} = \frac{P(c_T|c_I, c_V, z)P(c_I|c_V, z)P(c_V|z)P(z)}{P(c_T, c_I, c_V)}$$

Diffusion models estimate the score [9] of the data distribution, i.e., the derivative of the log probability. Taking the logarithm gives us the following expression:

$$\begin{aligned}\log(P(z|c_T, c_I, c_V)) = &\log(P(c_T|c_I, c_V, z)) \\ &+ \log(P(c_I|c_V, z)) \\ &+ \log(P(c_V|z)) + \log(P(z) \\ &- \log(P(c_T, c_I, c_V))\end{aligned}$$

Taking the derivative and rearranging we attain:

$$\begin{aligned}\nabla_z \log(P(z|c_T, c_I, c_V)) = &\nabla_z \log(P(z)) \\ &+ \nabla_z \log(P(c_V|z)) \\ &+ \nabla_z \log(P(c_i|c_V, z)) \\ &+ \nabla_z \log(P(c_T|c_I, c_V, z))\end{aligned}$$

This corresponds with the terms in our classifier-free guidance formulation in Equation 6.

G.4. Supporting Tasks for AnySD

In general, the editing tasks supported by AngSD align with those listed for AnyEdit-Test. For each task, we utilize a distinct learned task embedding of size N (where N matches the dimensionality of CLIP).

Additionally, there are substantial differences between various types of tasks. Consequently, we employ a Mixture of Experts (MoE) framework. Specifically, our expert categorization is detailed in Table 2.

Expert	Supporting tasks
Expert 1	tone transfer, background change, style transfer, style change
Expert 2	movement, outpaint, resize, rotation
Expert 3	visual bbox
Expert 4	visual depth
Expert 5	visual material transfer
Expert 6	visual reference
Expert 7	visual scribble
Expert 8	visual segment
Expert 9	visual sketch

Table 2. Expert division for various editing tasks of AnySD.

G.5. Training Details

Stage I: Instruction Understanding. In this stage, we use the dataset type of background change, tone transfer, remove, replace, add, color, and appearance change in AnyEdit to enhance the model's instruction-following capability. Following prior works [2, 33, 41], we train our image editing model for 110,000 steps using four 48GB NVIDIA A6000 GPUs for 280 hours. Specifically, the training is conducted at a resolution of 256 × 256 with a total batch size of 1024 (gradient_accumulation_steps=2, batch_size=128 per GPU). We apply random horizontal flip augmentation and crop augmentation, where images are first randomly resized between 256 and 288 pixels, followed by cropping to 256 × 256. The model is trained with a learning rate of 10^{-4}, without any learning rate warm-up. We initialize our model using the EMA weights from the Stable Diffusion 1.5 checkpoint [30] and adopt other training configurations from the publicly available Stable Diffusion codebase. Although the model is trained at a resolution of 256 × 256, it generalizes well to a resolution of 512 × 512 during inference. All results presented in this paper are generated at 512 × 512 resolution, with an Euler ancestral sampler and the denoising variance schedule proposed by [10].

Stage II: Task Tuning. In the second stage, we train our model on the entire AnyEdit dataset to adapt the model to the task-specific editing granularity. We utilize the task embedding and each expert is described in Appendix G.4. Unlike the first stage, we do not use EMA (Exponential Moving Average) for training [8]. Additionally, we set the training resolution to 512 × 512, compared to 256 × 256 in the first stage, to achieve better editing results for specific tasks. The model is trained with a learning rate of 10^{-4}, without any learning rate warm-up. The second stage is trained for 400,000 steps using eight 48GB NVIDIA A6000 GPUs over approximately 150 hours.

G.6. Baselines Details

We establish the models in Table 3 as baselines, organized into two categories: instruction-based and specific image editing methods. The former utilizes natural instructions to guide the editing process, while the latter relies on global descriptions of the target image to enable editing. Instruction-based image editing methods include InstructPix2Pix [2], HIVE [43], UltraEdit [44], EMU-Edit [33], and MagicBrush [11]. The specific image editing methods include Null Text Inversion [21], while the visual condition editing methods include Uni-ControlNet [45].

Instruction-Based Editing Methods:

- *InstructPix2Pix* [2]: Utilizes automatically generated instruction-based image editing data to fine-tune Stable Diffusion [29], enabling instruction-based image editing during inference without requiring any test-time tuning. We use the official Hugging Face to implement it.
- *HIVE* [43]: Trained with supplementary data akin to InstructPix2Pix, HIVE undergoes further fine-tuning using a reward model derived from human-ranked data. Notably, the edited output of HIVE is not square; instead, it is scaled to preserve the original aspect ratio, with the longer side resized to 512 pixels. We utilized two models, the weighted reward (SD1.5) and the conditional reward (SD1.5), referred to as **HIVE**^{w} and **HIVE**^{c}, respectively.
- *UltraEdit* [44]. It is trained on nearly 4 million instruction-based editing samples based on the Stable Diffusion 3 [32] and supports free-form and mask-form inputs to enhance editing performance. To ensure comparison fairness, we utilize its freeform model for all experiments. Notably, since it is trained on SD3, its performance cannot accurately reflect the improvements brought by its editing data.
- *EMU-Edit* [33]. It is a fine-tuned editing model that integrates recognition and generation tasks. Although it provides promising results, the model is not open-sourced. Therefore, we only conduct comparisons of it and AnyEdit on public benchmarks to demonstrate the superiority of our approach.
- *MagicBrush* [11]: MagicBrush curates a well-structured editing dataset with detailed human annotations and fine-tunes its model on this dataset using the InstructPix2Pix [2] framework. Therefore, we use this as a baseline to fairly compare the improvement in editing capabilities brought by the AnyEdit dataset in our experiments.

Specific Editing Methods:

- *Null Text Inversion* [21]: This method inverts the source image using the DDIM [34] trajectory and performs edits during the denoising process by controlling cross-attention between text and image. Notably, Null Text Inversion requires that "attention replacement editing can only be applied to prompts of the same length." Therefore, if the input and output captions differ in length, we align the word count by truncating the longer caption. Additionally, it is worth mentioning that the official repository performs a center crop when processing non-square images, and we adhered to this setting.

Visual Condition Methods:

- *Uni-controlnet* [45]. Uni-ControlNet categorizes conditions into two groups: local and global. By adding only two additional adapters, the cost of fine-tuning and the model size are significantly reduced. For local controls, we introduce a multi-scale conditional injection strategy, while for global controls, a global condition encoder is used to convert them into conditional tokens, which then interact with the incoming features. To let it support visual reference editing, we use the HED condition as the channel of reference image input.

Method	Configuration
InstructPix2Pix [2]	num_inference_steps=10, image_guidance_scale=1
MagicBrush [41]	seed=42, guidance_scale=7 num_inference_steps=20, image_guidance_scale=1.5
UltraEdit [44]	negative_prompt="", num_inference_steps=50, image_guidance_scale=1.5, guidance_scale=7.5
HIVE^w [43]	steps=100 text_cfg_scale=7.5, image_cfg_scale=1.5
HIVE^c [43]	steps=100 text_cfg_scale=7.5, image_cfg_scale=1.5
Null-Text [21]	cross_replace_steps.default=0.8, self_replace_steps= 0.5, blend_words=None, equilizer_params=None
Uni-controlnet [45]	num_samples = 1 image_resolution = 512 strength = 1 global_strength = 1 low_threshold = 100 high_threshold = 200 value_threshold = 0.1 distance_threshold = 0.1 alpha = 6.2 ddim_steps = 50 scale = 7.5 seed = 42 eta = 0.0 a_prompt = 'best quality, extremely detailed' n_prompt = 'longbody, lowres, bad anatomy, bad hands, missing fingers, extra digit, fewer digits, cropped, worst quality, low quality'

Table 3. **Configuration of the baselines for AnyEdit-Test**. We strictly adhered to the default hyperparameters provided in the official repositories or Huggingface implementations of these baseline models.

G.7. Details on Benchmarks and Metrics

Metrics and code. For metrics evaluation, we closely follow the MagicBrush evaluation script without any modifications. Following previous works [1, 41, 44], we employ L1 metrics to measure pixel-level differences between the

generated and ground truth images. Additionally, CLIP and DINO similarities are used to assess the overall similarity with the ground truth, while CLIP-T measures text-image alignment based on local descriptions and the CLIP embedding of generated images. Furthermore, CLIP text-image similarity between the edited image and the output caption, as well as CLIP text-image direction similarity (CLIPdir), are employed to evaluate the model's instruction-following ability. Specifically, CLIPdir measures the agreement between changes in caption embedding and changes in image embedding. While the Emu Edit Test eliminates bias and overfitting at the image level by not providing ground truth images, the evaluation metrics still implicitly assess the model's editing capabilities.

EMU-Edit-Test. We observe that the original EMU-Edit [33] paper and dataset don't specify the versions of CLIP [25] and DINO [3] used. To maintain consistency with other benchmarks, we follow the settings from the MagicBrush repository [41], modifying only the evaluation dataset to EMU-Edit-Test.

MagicBrush-Test. MagicBrush is designed to evaluate both the single-turn and multi-turn image editing capabilities of models. It provides annotator-defined instructions and editing masks, along with ground truth images generated by DALLE-2 [26], facilitating a more effective metric-based assessment of the model's editing performance. However, the dataset suffers from inherent biases. During data collection, annotators are instructed to use the DALLE-2 image editing platform to generate the edited images, making the benchmark biased towards images and editing instructions that the DALLE-2 editor can successfully follow. This bias may limit the dataset's diversity and complexity. The baseline results in Table 4 of the main paper correspond to EMU-Edit [33].

H. Qualitative and Human Evaluations

H.1. Human Evaluation

We conduct comprehensive human evaluations to assess both the consistency and image quality of generated images across three tasks: multiple-choice comparison, pairwise comparison, and individual image assessment. For each task, we randomly sample 100 images from AnyEdit-Test (excluding the visual instruction component). These images are evenly distributed among evaluators, and where applicable, we report averaged scores. Specifically, we evaluate three methods, comparing our approach against four SOTA editing methods: InstructPix2Pix [2], MagicBrush [41], HIVE^w [43], HIVE^c [43], UltraEdit (SD3) [44] and our method.

Multiple-Choice Comparison. In this task, evaluators select the best-edited image based on consistency and image quality. As shown in Table 4, our method demonstrates superior performance, significantly surpassing the other methods, which emphasizes the effectiveness of training on our AnyEdit dataset. Notably, while MagicBrush and UltraEdit score highly in automated evaluations, their performance in human assessments is comparatively lower, especially in instruction consistency. This discrepancy highlights the limitations of current automatic metrics, which focus primarily on image quality and may not fully capture human preferences, underscoring the need for future research to develop more robust and aligned evaluation metrics.

	Consistency	Image Quality
MagicBrush [41]	10	9
HIVE^w [43]	15	17
HIVE^c [43]	20	21
UltraEdit (SD3) [44]	13	17
AnySD	**42**	**36**

Table 4. Multi-choice comparison of four methods. The numbers represent the frequency of each method being chosen as the best for each aspect.

	Consistency	Image Quality
MagicBrush [41]	0.35	0.27
HIVE^w [43]	0.42	0.41
HIVE^c [43]	0.47	0.48
UltraEdit (SD3) [44]	0.35	0.37

Table 5. One-on-one comparisons. The numbers in the table indicate the percent of each method being chosen as the better option compared with the AnySD's results.

One-on-One Comparison.The one-on-one comparison provides a detailed and nuanced assessment of the edited results by juxtaposing them with robust baselines. Evaluators are instructed to select the preferred option based on both consistency and image quality. As shown in Table 5, AnySD consistently outperforms the alternatives in both aspects, with a majority of evaluators favoring AnySD's results in these direct comparisons.

Individual Evaluation. The individual evaluation utilizes a 5-point Likert scale to gather subjective feedback on image quality generated by four distinct models. Evaluators rate each image from 1 to 5, focusing on both consistency and overall quality. As shown in Table 6, the results clearly indicate that AnySD outperforms the other baselines, underscoring the advantages of training or fine-tuning models on the AnyEdit dataset.

H.2. Qualitative Evaluation on Different Benchmarks

Detailed Results for EMU-Edit Test. More qualitative results of the EMU-Edit Test are shown in Figure 9. We observe that AnySD can effectively distinguish between the foreground and background of an image solely based on

	Consistency	Image Quality
MagicBrush [41]	3.3	3.1
HIVE^{w} [43]	4.1	3.8
HIVE^{c} [43]	4.2	4.2
UltraEdit (SD3) [44]	3.7	4.0
AnySD	**4.3**	**4.4**

Table 6. Individual evaluation using a 5-point Likert scale. The numbers in the table represent the average scores calculated for each aspect.

editing instructions, accurately modifying the background while preserving the content of the foreground.

Detailed Results for MagicBrush Benchmark. More qualitative results of the MagicBrush Test are shown in Figure 10. We visually compare the performance of our method on local editing with the SOTA mask-based Editing model (*i.e.*, DALLE-2 [26]). We notice that even without masks as supervision signals, our method accurately performs edits in specific regions, benefiting from the well-aligned editing data provided by AnyEdit.

Detailed Results for AnyEdit-Test. More qualitative results of the AnyEdit-Test are shown in Figure 11, 12, 13.

More qualitative results of high-quality image editing from AnySD. In Figure 14, we visualize AnySD editing results on a wide variety of images. We provide different editing instructions for the same image and observe that our method consistently achieves high-quality and fine-grained editing. For example, it successfully modifies underwater reflections and performs appearance modifications involving world knowledge. It effectively demonstrates the high quality of the AnyEdit dataset and the superiority of the AnySD architecture.

Multi-turn in MagicBrush. Figures 15 and 16 illustrate the performance of our AnySD model in multi-turn editing. Compared to Text2LIVE [36], GLIDE [24], InstructPix2Pix, and MagicBrush, our model demonstrates stronger consistency, maintaining greater similarity to the original image even in the final editing rounds. Our results even surpass the ground truth provided by MagicBrush, further affirming the high quality of the AnyEdit dataset.

Additional Image Editing Methods. We also evaluate our image editing model in comparison with other approaches, including Versatile Diffusion [37], BLIP Diffusion [14], Uni-ControlNet [45], T2I-Adapter [22], ControlNet Shuffle [42], ControlNet Reference-only [42], and IP-Adapter [39]. The comparison results are presented in Figure 17, 18. Compared to other methods, our approach consistently produces superior results in terms of image quality and alignment with multimodal prompts.

Type	Description
Local Editing	
Remove	Remove a specific object in the image and fill it with a background.
Replace	Replace a specific object in the image with a new object.
Add	Inserting a new object in the image.
Counting	Removing a specified number of objects to satisfy the number requirement.
Color Alter	Altering the color of specific objects in the image.
Appearance Alter	Altering the appearance (*e.g.*, decoration, texture, illumination) of specific objects in the image.
Action Change	Change the action of the specific object in the image.
Textual Change	Change the specific textual contents in the image to new textual contents
Material Change	Change the material of the specific object in the image.
Global Editing	
Background Change	Modifying the background of the entire image but not affecting the foreground objects.
Tone Transfer	Modifying the overall tone of the image, including changes in time, weather, and seasons.
Style Change	Modifying the overall style of the image according to the given style word.
Camera Movement Editing	
Movement	Move the position viewpoint of a specific object in the image to the left or right or up or down.
Resize	Zoom in or zoom out to a specific object in the image.
Outpainting	Expanding the overall viewpoint by imagining the surroundings of the image elements of the mask.
Rotation Change	Rotate the overall viewpoint of the image to obtain images from different perspectives.
Implicit Editing	
Implicit Change	Implicitly altering the contents of an image necessitates comprehension rather than explicit instructions.
Relation Change	Change the position relationship between two objects (*i.e.*, swap their positions) in the image.
Visual Editing	
Image Reference	Replace a specific object in the image with the object in the reference image instead of any word.
Material Transfer	Transfer the material in the reference image to the specific object in the image.
Style Transfer	Transfer the style in the reference image to the specific image.
Visual Bounding Box	Utilizing bounding box images as visual conditions images to guide the removal or replacement.
Visual Scribble	Utilizing scribble images as visual conditions images to guide the removal or replacement.
Visual Segmentation	Utilizing segmentation images as visual conditions images to guide the removal or replacement.
Visual Sketch	Utilizing sketch images as visual conditions images to guide the removal or replacement.
Visual Depth	Utilizing depth images as visual conditions images to guide the removal or replacement.

Table 7. The Definition of Editing Instruction Types in AnyEdit.

Element	Content
System Prompt	You are an assistant that only speaks JSON. Do not write normal text. The assistant answer is JSON with the following string fields: `edit`, `edited object`, `output`. Here is the latest conversation between the Assistant and the User.
Task Description (Model)	Hi, My job is to take a given caption (`input`) and to output the following: an instruction for adding an object to the image (`edit`), the object to add (`edited object`), and the caption with the object (`output`). Please help me do it. I will give you the `input`, and you will help.
Output Format	When you reply, use the following format: {`'edit': '<instruction>', 'edited object': '<object>', 'output': '<caption>'`}. Construct the instruction with one of the following instruction words: [`place`, `add`, `include`]. Don't include any \ or edit any actions in the instruction.
Initial Message from LLM	Sure, I'd be happy to help! Just provide me with the `input` (the original caption), and I'll generate the instruction, edited object, and output caption for you. Let's get started!
Example Input	User input: Beautiful cat with mojito sitting in a cafe on the street.
Example Output from LLM	{'edit': 'add a hat to the cat', 'edited object': 'hat', 'output': 'Beautiful cat wearing a hat with mojito sitting in a cafe on the street.'}
Constrained Editing Instruction Generation	
Input	{`caption`} from real-world image-text pairs or counterfactual synthetic datasets with {`editing type`} requirements from users
Response	{`model generated edit instructions in JSON format`}. An example of editing data is: { "edit": "change the airplane to green", "edited object": "airplane", "input": "a small airplane sits stationary on a piece of concrete.", "output": "A green small airplane sits stationary on a piece of concrete.", "edit type": "color alter", "visual input": "None", "image file": "COCO-train2014-000000521165.jpg", "edited file": "xxxxx.jpg" }"

Table 8. Prompts constraints for LLMs to write edit instructions and captions.

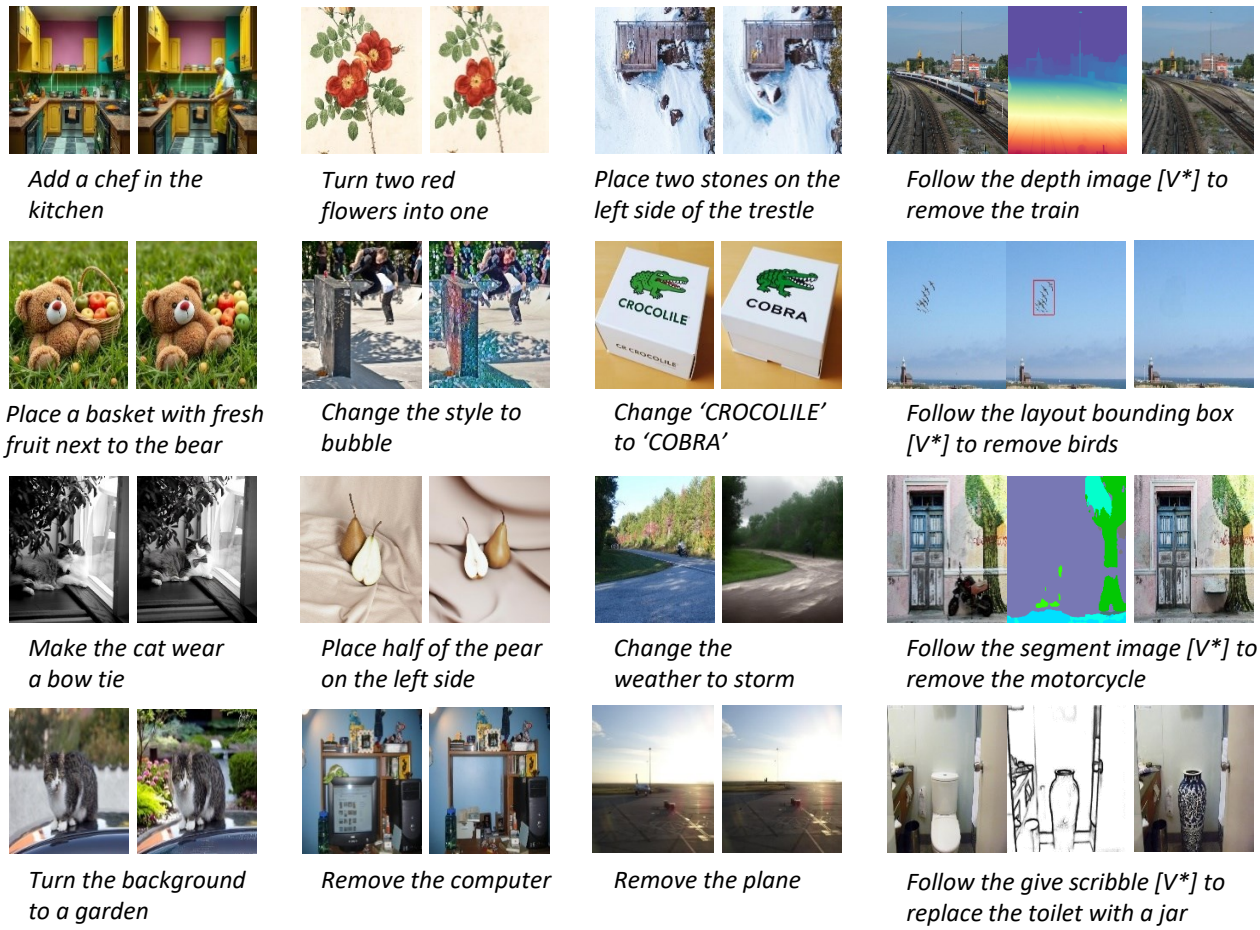
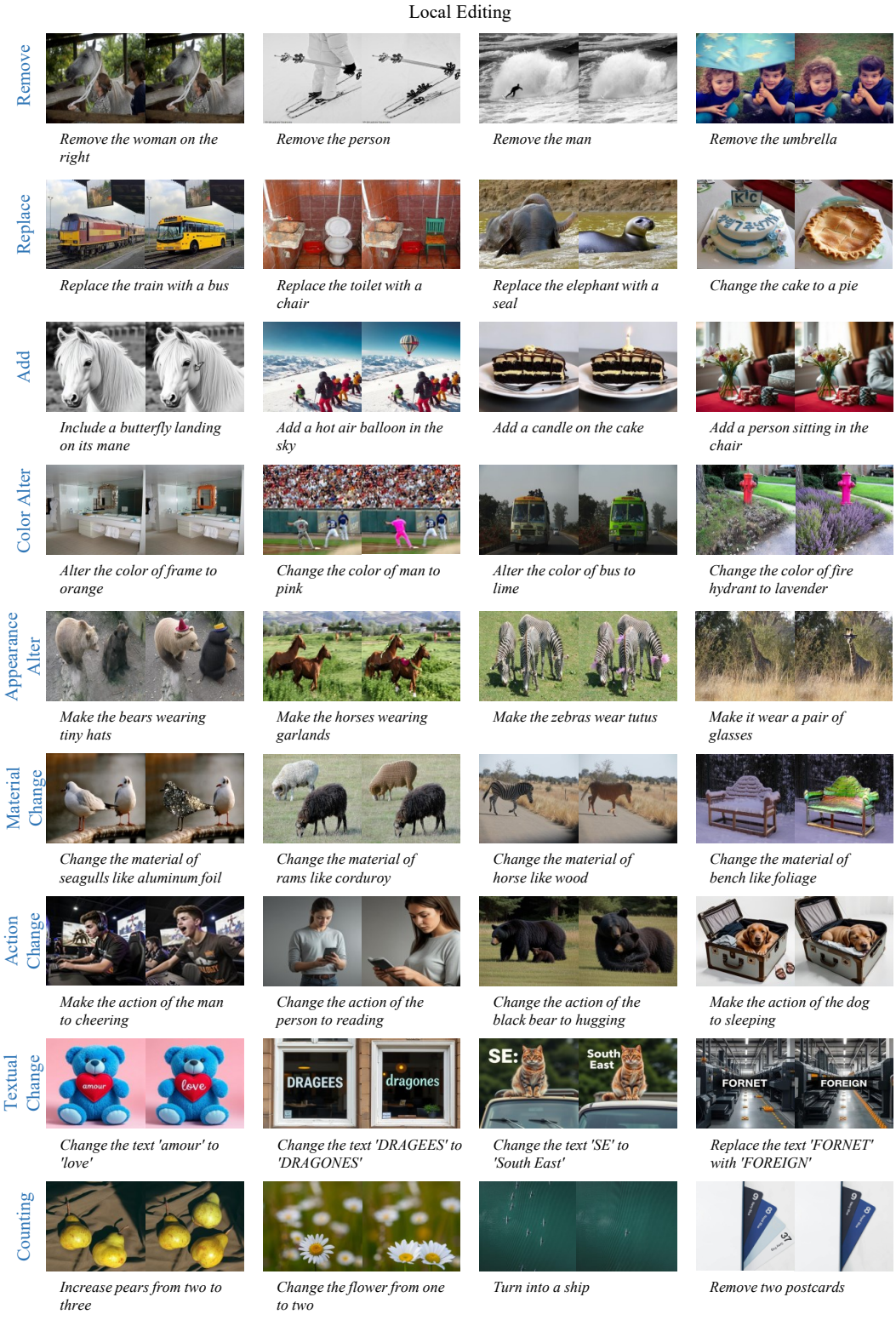


Figure 2. More Examples of AnyEdit dataset (Part 1). textual instruction-based (first three columns) and visual instruction-based (last column) image editing.

Figure 3. More Examples of AnyEdit dataset (Part 2). textual instruction-based (first three columns) and visual instruction-based (last column) image editing.

Figure 4. More Examples of AnyEdit-Test with local editing categories.



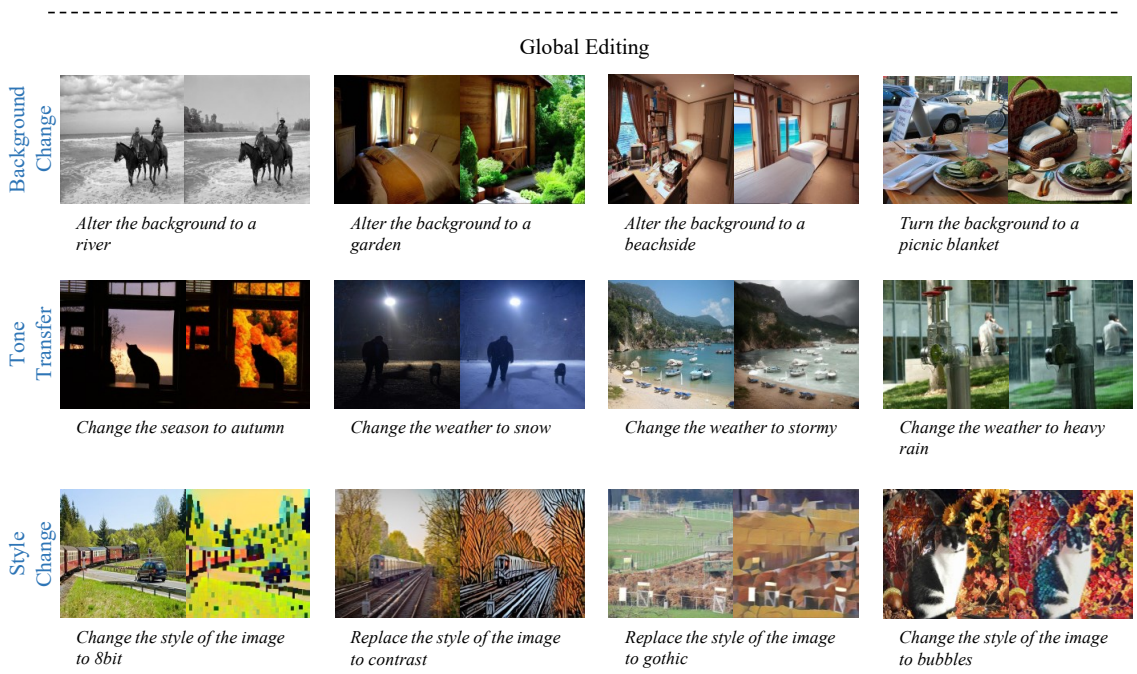


Figure 5. More Examples of AnyEdit-Test with global editing categories.

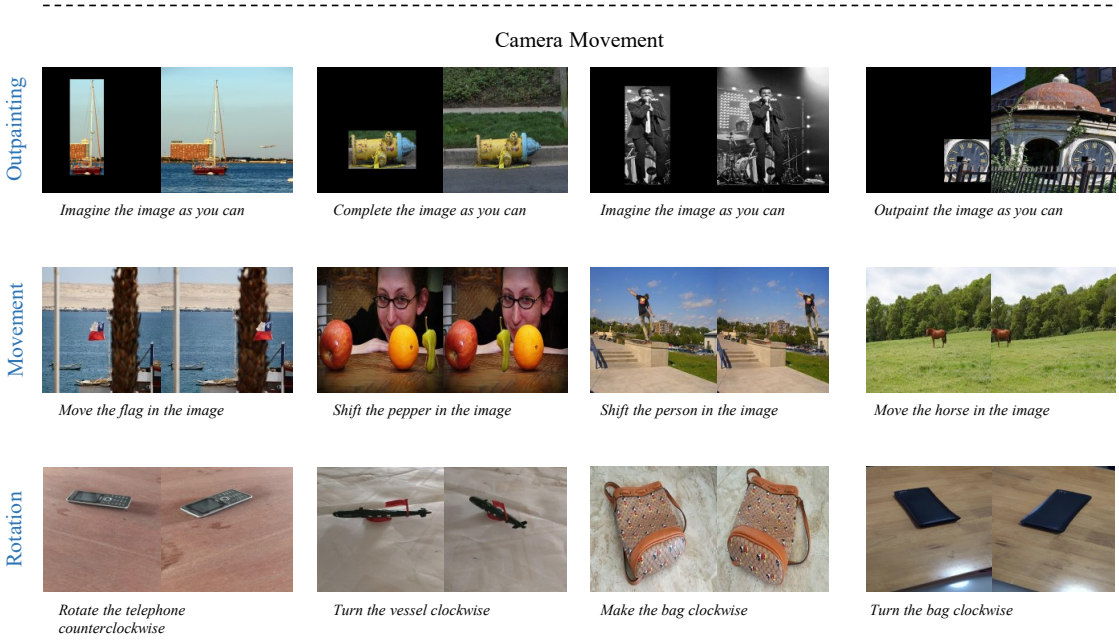


Figure 6. More Examples of AnyEdit-Test with camera movement editing categories.

Figure 7. More Examples of AnyEdit-Test with visual editing categories.

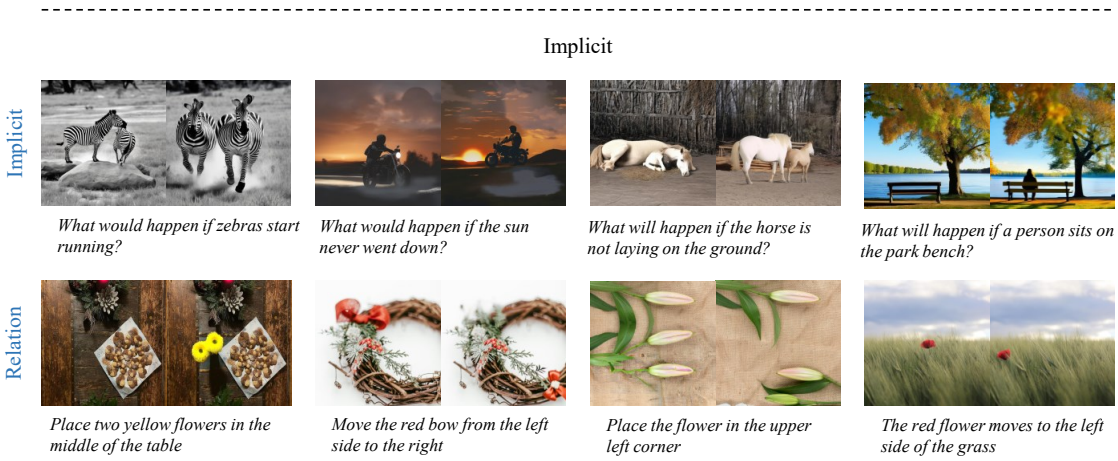


Figure 8. More Examples of AnyEdit-Test with implicit editing categories.

	local								
	remove	replace	add	color	appearance	material change	action	textual	counting
InstructPix2Pix [2]									
CLIPim ↑	0.664	0.779	0.832	0.862	0.770	0.700	0.674	0.744	0.803
CLIPout ↑	0.227	0.276	0.302	0.318	0.308	-	0.228	0.298	0.272
L1 ↓	0.146	0.188	0.134	0.162	0.160	0.168	0.167	0.190	0.149
DINO ↑	0.408	0.537	0.706	0.773	0.593	0.369	0.413	0.694	0.590
MagicBrush [41]									
CLIPim ↑	0.849	0.814	0.930	0.826	0.843	0.809	0.754	0.759	0.875
CLIPout ↑	0.264	0.289	0.321	0.305	0.319	-	0.272	0.312	0.264
L1 ↓	**0.076**	0.143	0.071	0.112	0.084	0.111	0.203	0.157	0.100
DINO ↑	0.783	0.604	0.897	0.667	0.739	0.570	0.548	0.774	0.731
HIVE^{w} [43]									
CLIPim ↑	0.750	0.788	0.914	0.853	0.819	0.764	0.826	0.801	0.866
CLIPout ↑	0.237	0.282	0.312	0.307	0.313	-	0.291	0.318	0.266
L1 ↓	0.118	0.184	0.079	0.114	0.147	0.126	0.155	0.139	0.122
DINO ↑	0.586	0.600	0.857	0.779	0.690	0.536	0.735	0.838	0.738
HIVE^{c} [43]									
CLIPim ↑	0.823	0.778	0.932	**0.894**	0.864	0.785	**0.874**	0.807	**0.899**
CLIPout ↑	0.254	0.284	0.312	0.309	0.309	-	**0.308**	0.319	0.267
L1 ↓	0.099	0.167	0.066	0.097	0.105	0.103	0.147	0.129	0.100
DINO ↑	0.728	0.584	0.891	0.850	0.795	0.594	**0.811**	0.871	0.800
UltraEdit (SD3) [44]									
CLIPim ↑	0.806	0.805	0.925	0.851	0.817	0.764	0.827	**0.854**	0.880
CLIPout ↑	0.262	**0.295**	0.323	**0.320**	**0.320**	-	0.292	**0.344**	**0.273**
L1 ↓	0.087	0.151	0.072	0.091	0.100	0.108	0.158	**0.127**	0.089
DINO ↑	0.709	0.615	0.867	0.791	0.729	0.522	0.724	**0.890**	0.764
Null-Text [21]									
CLIPim ↑	0.752	0.710	-	0.814	0.785	-	0.838	0.764	-
CLIPout ↑	0.250	0.247	-	0.274	0.285	-	0.298	0.305	-
L1 ↓	0.235	0.253	-	0.227	0.239	-	0.243	0.275	-
DINO ↑	0.598	0.384	-	0.695	0.675	-	0.732	0.764	-
AnySD w/ AnyEdit (Ours)									
CLIPim ↑	**0.851**	**0.853**	**0.946**	0.896	**0.877**	**0.811**	0.873	0.763	0.898
CLIPout ↑	**0.265**	0.292	**0.322**	0.313	0.309	-	0.306	0.303	0.263
L1 ↓	0.103	**0.123**	**0.052**	**0.061**	**0.051**	**0.084**	**0.145**	0.136	**0.088**
DINO ↑	**0.785**	**0.688**	**0.921**	**0.855**	**0.840**	**0.602**	0.782	0.800	**0.819**

Table 9. **Comparison of Methods on AnyEdit-Test (Part 1)**. '-' indicates 'not applicable'.

	global			camera				implicit	
	background	tone transfer	style change	movement	outpaint	rotation	resize	implicit	relation
InstructPix2Pix [2]									
CLIPim ↑	0.680	**0.860**	0.702	0.805	0.563	0.675	0.755	0.762	0.826
CLIPout ↑	0.259	**0.304**	-	-	-	-	-	-	0.288
L1 ↓	0.221	**0.098**	0.221	0.131	0.290	0.148	0.141	0.212	0.167
DINO ↑	0.411	0.804	0.354	0.639	0.341	0.361	0.566	0.538	0.577
MagicBrush [41]									
CLIPim ↑	0.739	0.789	0.664	0.863	0.561	0.791	0.845	0.819	0.910
CLIPout ↑	0.268	0.287	-	-	-	-	-	-	0.280
L1 ↓	0.233	0.213	0.252	0.093	0.353	0.134	0.101	0.189	0.109
DINO ↑	0.529	0.657	0.292	0.710	0.344	0.575	0.725	0.622	0.800
HIVE^{w} [43]									
CLIPim ↑	0.764	0.816	0.706	0.872	0.582	0.774	0.888	0.784	0.858
CLIPout ↑	0.280	0.293	-	-	-	-	-	-	0.284
L1 ↓	0.202	0.175	0.212	0.131	0.328	0.135	0.107	0.202	0.119
DINO ↑	0.635	0.719	0.383	0.732	0.328	0.620	0.796	0.572	0.697
HIVE^{c} [43]									
CLIPim ↑	**0.822**	0.833	0.705	**0.926**	0.665	**0.848**	**0.912**	0.809	**0.914**
CLIPout ↑	0.294	0.293	-	-	-	-	-	-	0.284
L1 ↓	0.177	0.182	0.401	0.112	0.349	0.129	0.093	0.180	0.093
DINO ↑	**0.777**	0.748	0.202	**0.866**	0.428	**0.739**	**0.861**	0.627	**0.829**
UltraEdit (SD3) [44]									
CLIPim ↑	0.790	0.795	**0.730**	0.867	**0.705**	0.765	0.872	**0.825**	0.887
CLIPout ↑	0.293	0.301	-	-	-	-	-	-	0.281
L1 ↓	0.181	0.184	0.208	0.106	0.372	0.139	0.086	0.176	0.093
DINO ↑	0.701	0.709	0.448	0.762	0.612	0.523	0.813	0.642	0.764
Null-Text [21]									
CLIPim ↑	0.755	0.750	-	-	-	-	-	-	-
CLIPout ↑	0.285	0.269	-	-	-	-	-	-	-
L1 ↓	0.251	0.289	-	-	-	-	-	-	-
DINO ↑	0.617	0.608	-	-	-	-	-	-	-
AnySD w/ AnyEdit (Ours)									
CLIPim ↑	0.819	0.836	0.710	0.870	0.738	0.826	0.898	**0.825**	0.908
CLIPout ↑	**0.300**	0.302	-	-	-	-	-	-	**0.289**
L1 ↓	**0.169**	0.115	**0.192**	**0.069**	**0.189**	**0.122**	**0.060**	**0.169**	**0.091**
DINO ↑	0.744	**0.811**	0.385	0.782	**0.682**	0.685	0.832	**0.643**	0.822

Table 10. **Comparison of Methods on AnyEdit-Test (Part 2)**. '-' indicates 'not applicable'.

	Visual						
	visual depth	visual sketch	visual scribble	visual segment	visual bbox	material transfer	visual reference
Uni-controlnet [45]							
CLIPim ↑	0.741	0.763	0.770	0.716	0.734	0.642	0.652
CLIPout ↑	0.246	0.259	0.253	0.246	0.253	-	0.234
L1 ↓	0.271	0.247	0.254	0.281	0.214	0.278	0.275
DINO ↑	0.503	0.576	0.531	0.421	0.512	0.241	0.308
AnySD w/ AnyEdit (Ours)							
CLIPim ↑	**0.780**	**0.803**	**0.805**	**0.770**	**0.811**	**0.849**	**0.714**
CLIPout ↑	**0.250**	**0.268**	**0.258**	**0.252**	**0.258**	-	**0.260**
L1 ↓	**0.177**	**0.164**	**0.158**	**0.181**	**0.125**	**0.090**	**0.121**
DINO ↑	**0.612**	**0.663**	**0.627**	**0.607**	**0.687**	**0.712**	**0.488**

Table 11. **Comparison of Methods on AnyEdit-Test (Part 3)**. '-' indicates 'not applicable'.

Figure 9. More qualitative results of the EMU-Edit Test for the editing of background change.

Figure 10. More qualitative results of the MagicBrush Test for local editing. The mask is used solely to supervise the editing process in DALLE-2 [26] and is not provided as input to our method.

Figure 11. More qualitative evaluation of our model trained on AnyEdit across AnyEdit-Test benchmark (Part I).

Figure 12. More qualitative evaluation of our model trained on AnyEdit across AnyEdit-Test benchmark (Part II).

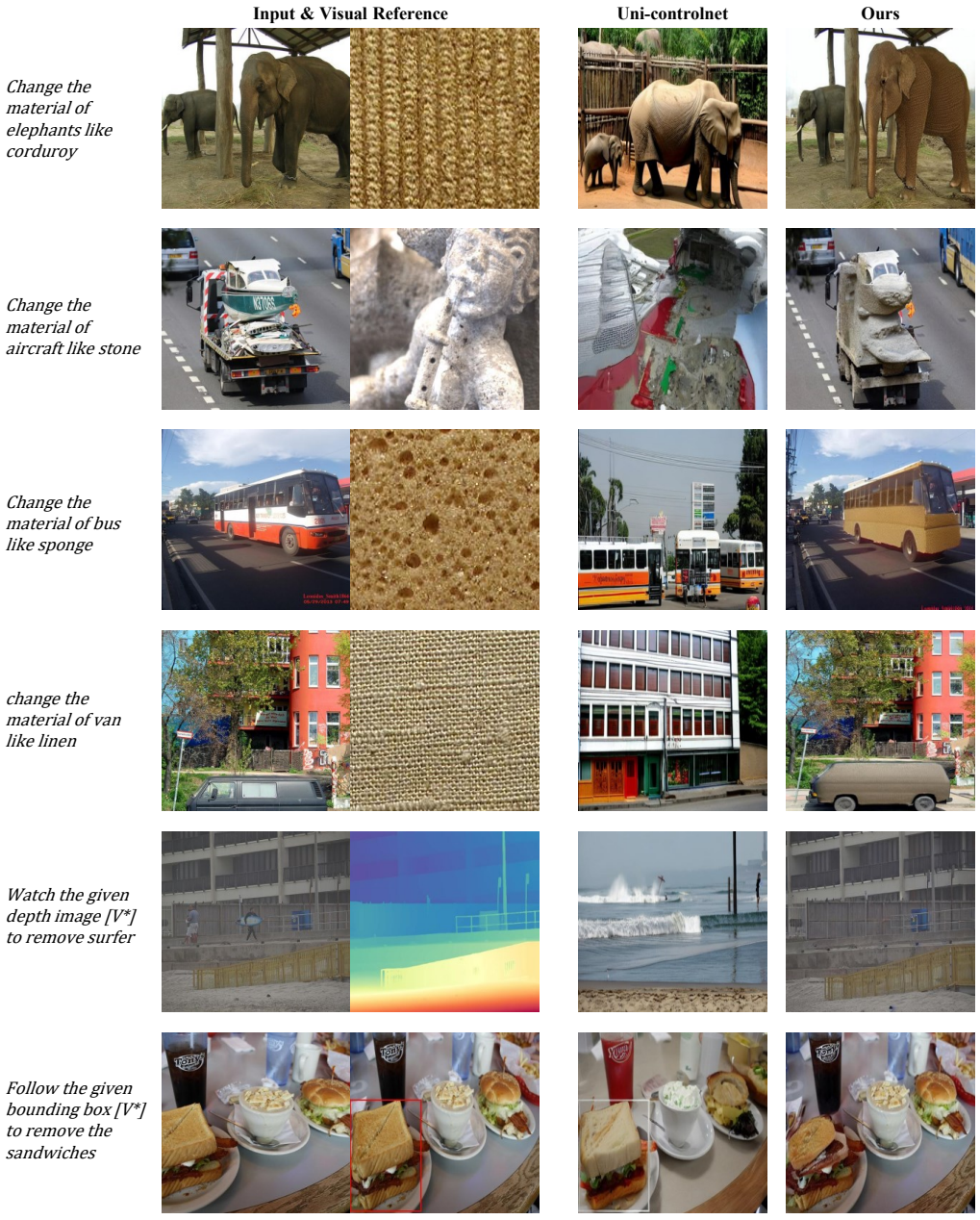


Figure 13. More qualitative evaluation of our model trained on AnyEdit across AnyEdit-Test benchmark (Part III).

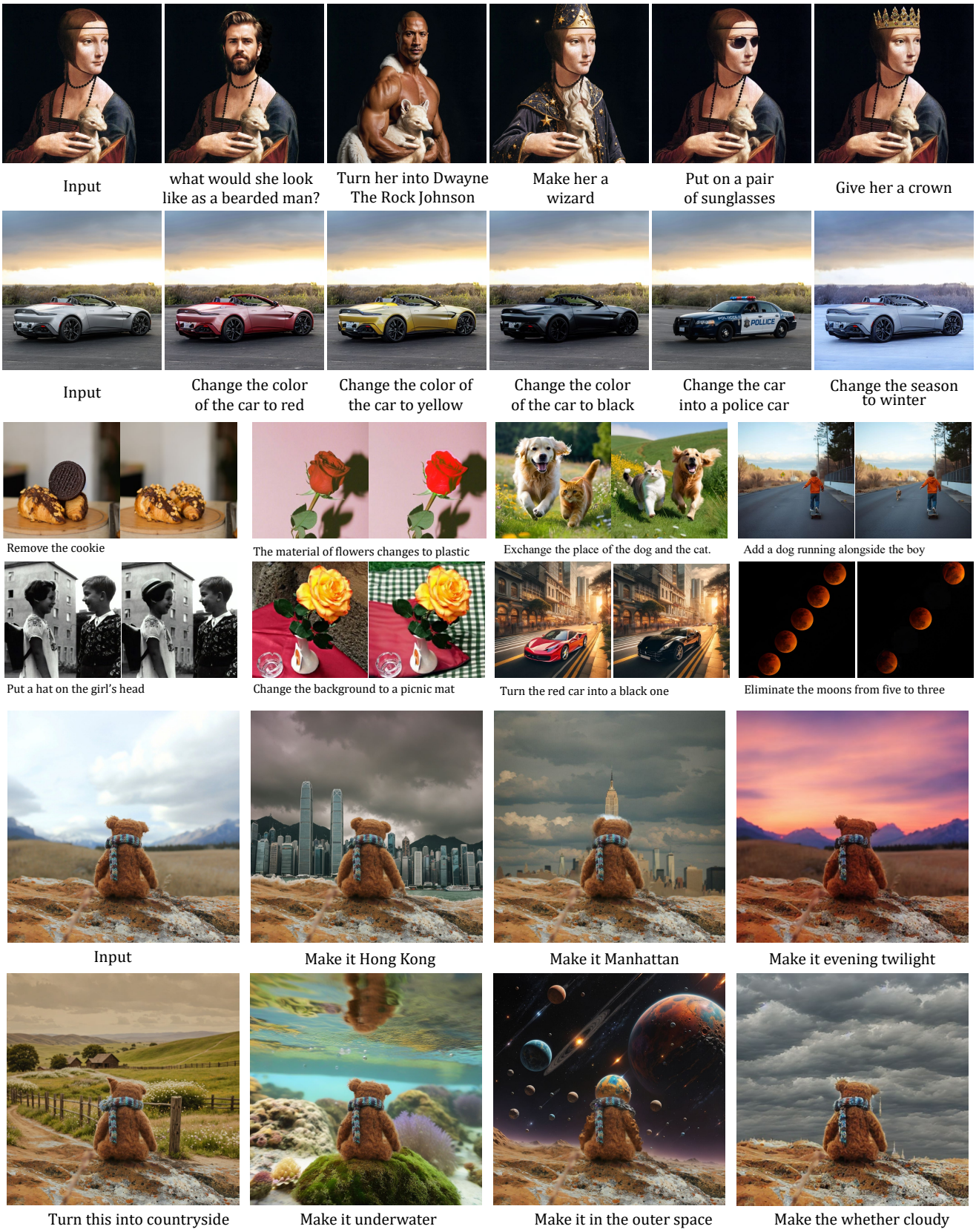


Figure 14. Qualitative evaluation of using real images as user inputs for the robustness of our editing model.

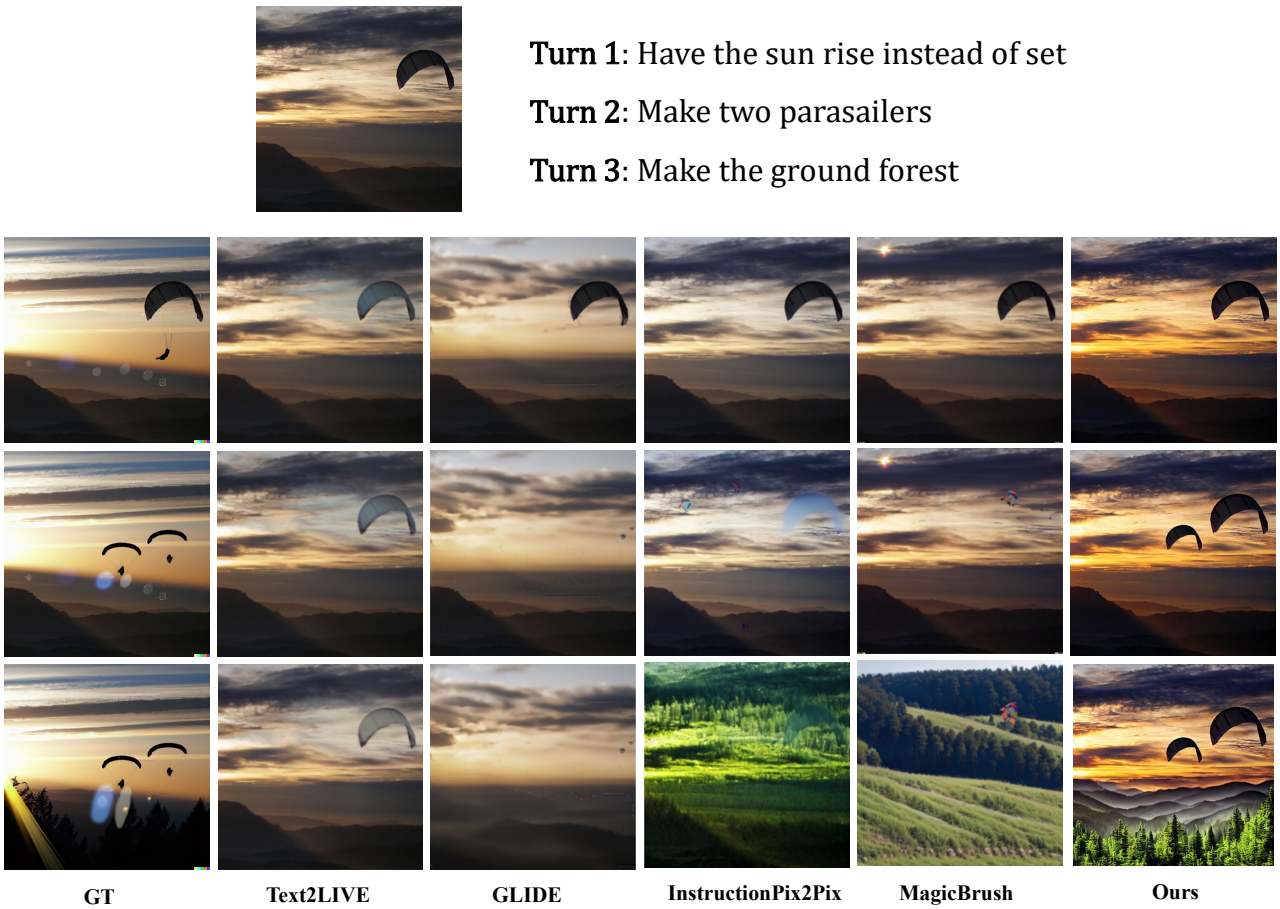


Figure 15. Qualitative evaluation of multi-turn editing scenario. We provide all baselines their desired input formats (Part I).

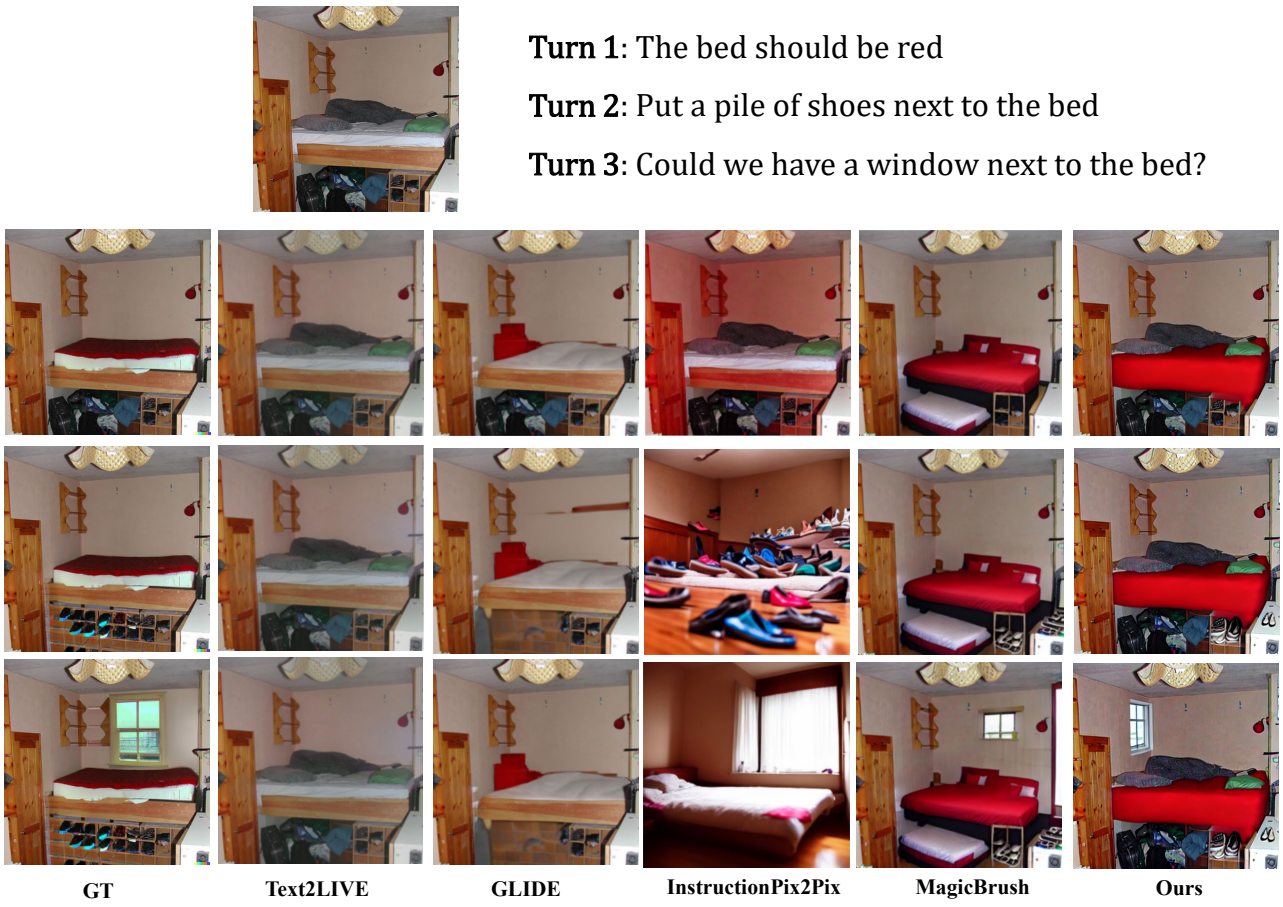


Figure 16. Qualitative evaluation of multi-turn editing scenario. We provide all baselines their desired input formats (Part II).

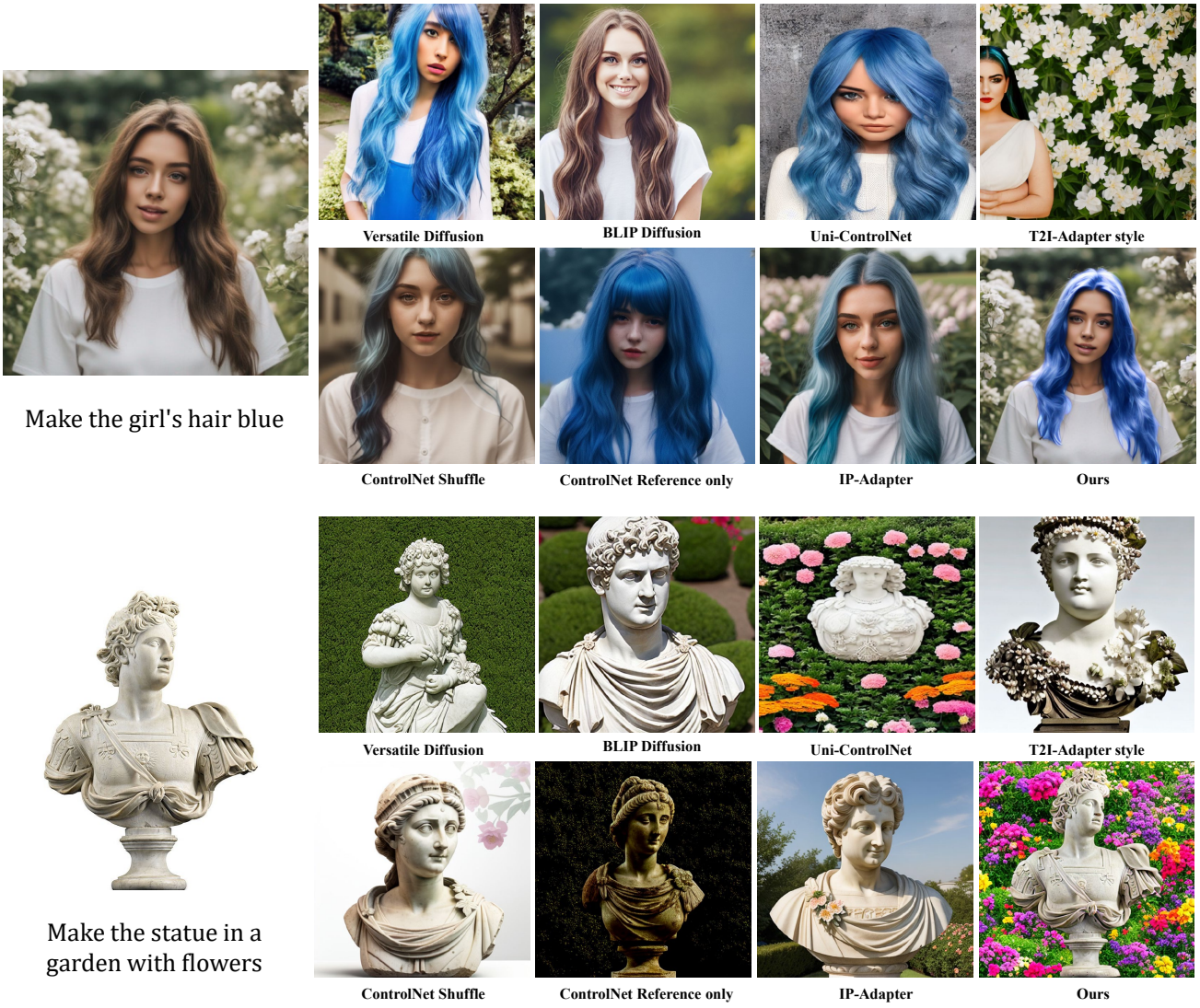


Figure 17. Comparison with more other image instruction edit methods (Part I).

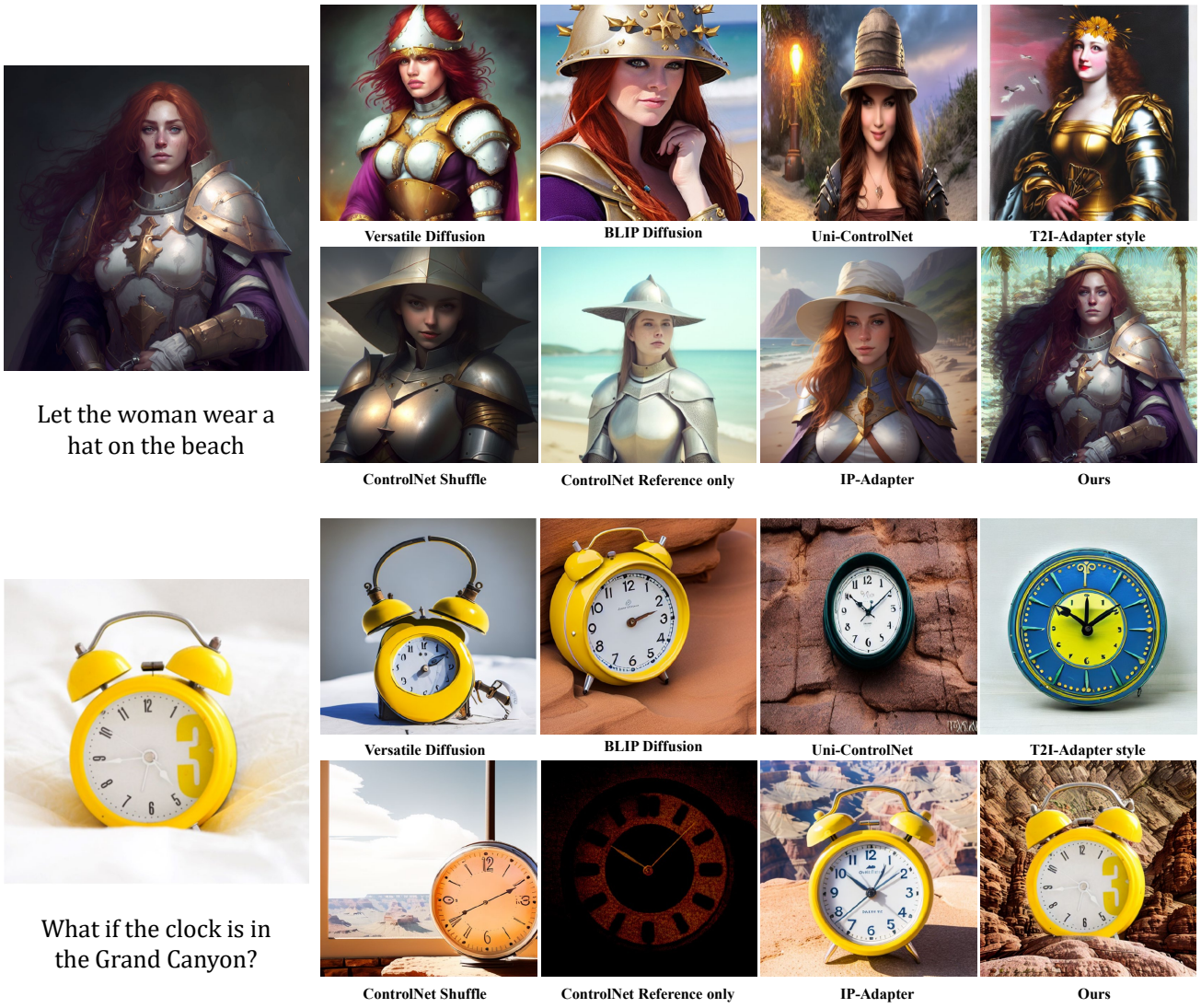


Figure 18. Comparison with more other image instruction edit methods (Part II).

References

[1] Jinbin Bai, Tian Ye, Wei Chow, Enxin Song, Qing-Guo Chen, Xiangtai Li, Zhen Dong, Lei Zhu, and Shuicheng Yan. Meissonic: Revitalizing masked generative transformers for efficient high-resolution text-to-image synthesis. *arXiv preprint arXiv:2410.08261*, 2024. 7

[2] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023. 2, 3, 5, 6, 7, 8, 18, 19

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 8

[4] Xi Chen, Lianghua Huang, Yu Liu, Yujun Shen, Deli Zhao, and Hengshuang Zhao. Anydoor: Zero-shot object-level image customization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6593–6602, 2024. 3

[5] Chee Kheng Chng, Yuliang Liu, Yipeng Sun, Chun Chet Ng, Canjie Luo, Zihan Ni, ChuanMing Fang, Shuaitao Zhang, Junyu Han, Errui Ding, et al. Icdar2019 robust reading challenge on arbitrary-shaped text-rrc-art. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1571–1576. IEEE, 2019. 2

[6] COCO-Text. A large-scale scene text dataset based on mscoco. https://bgshih.github.io/cocotext, 2016. 2

[7] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 5

[8] J Stuart Hunter. The exponentially weighted moving average. *Journal of quality technology*, 18(4):203–210, 1986. 6

[9] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005. 5, 6

[10] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022. 6

[11] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6007–6017, 2023. 6, 7

[12] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 5

[13] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 1

[14] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023. 9

[15] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*, pages 423–439. Springer, 2022. 5

[16] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 1

[17] LSVT. Icdar2019 robust reading challenge on large-scale street view text with partial labeling. https://rrc.cvc.uab.es/?ch=16, 2019. 2

[18] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42: 275–293, 2014. 5

[19] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Marcus A Brubaker, Jonathan Kelly, Alex Levinshtein, Konstantinos G Derpanis, and Igor Gilitschenski. Watch your steps: Local image and scene editing by text instructions. In *European Conference on Computer Vision*, pages 111–129. Springer, 2025. 2

[20] MLT. Icdar 2019 robust reading challenge on multi-lingual scene text detection and recognition. https://rrc.cvc.uab.es/?ch=15, 2019. 2

[21] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6038–6047, 2023. 6, 7, 18, 19

[22] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4296–4304, 2024. 9

[23] MTWI. Icpr 2018 challenge on multi-type web images. https://tianchi.aliyun.com/dataset/137084, 2018. 2

[24] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021. 9

[25] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 8

[26] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1 (2):3, 2022. 8, 9, 21

[27] RCTW. Icdar2017 competition on reading chinese text in the wild. https://rctw.vlrlab.net/dataset, 2017. 2

[28] ReCTS. Icdar 2019 robust reading challenge on reading chinese text on signboard. https://rrc.cvc.uab.es/?ch=12, 2019. 2

[29] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image

synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1, 5, 7

[30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 5, 6

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 5

[32] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. *arXiv preprint arXiv:2403.12015*, 2024. 7

[33] Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu edit: Precise image editing via recognition and generation tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8871–8879, 2024. 6, 7, 8

[34] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 5, 7

[35] Yuxiang Tuo, Wangmeng Xiang, Jun-Yan He, Yifeng Geng, and Xuansong Xie. Anytext: Multilingual visual text generation and editing. 2023. 2

[36] Chao Xu, Jiangning Zhang, Yue Han, Guanzhong Tian, Xianfang Zeng, Ying Tai, Yabiao Wang, Chengjie Wang, and Yong Liu. Designing one unified framework for high-fidelity face reenactment and swapping. In *European conference on computer vision*, pages 54–71. Springer, 2022. 9

[37] Xingqian Xu, Zhangyang Wang, Gong Zhang, Kai Wang, and Humphrey Shi. Versatile diffusion: Text, images and variations all in one diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7754–7765, 2023. 9

[38] Ling Yang, Bohan Zeng, Jiaming Liu, Hong Li, Minghao Xu, Wentao Zhang, and Shuicheng Yan. Editworld: Simulating world dynamics for instruction-following image editing. *arXiv preprint arXiv:2405.14785*, 2024. 3

[39] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. 2023. 9

[40] Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Chenming Zhu, Zhangyang Xiong, Tianyou Liang, et al. Mvimgnet: A large-scale dataset of multi-view images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9150–9161, 2023. 3

[41] Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. Magicbrush: A manually annotated dataset for instruction-guided image editing. *Advances in Neural Information Processing Systems*, 36, 2024. 6, 7, 8, 9, 18, 19

[42] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 3, 9

[43] Shu Zhang, Xinyi Yang, Yihao Feng, Can Qin, Chia-Chih Chen, Ning Yu, Zeyuan Chen, Huan Wang, Silvio Savarese, Stefano Ermon, Caiming Xiong, and Ran Xu. Hive: Harnessing human feedback for instructional visual editing. *arXiv preprint arXiv:2303.09618*, 2023. 6, 7, 8, 9, 18, 19

[44] 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. *arXiv preprint arXiv:2407.05282*, 2024. 6, 7, 8, 9, 18, 19

[45] Shihao Zhao, Dongdong Chen, Yen-Chun Chen, Jianmin Bao, Shaozhe Hao, Lu Yuan, and Kwan-Yee K. Wong. Uni-controlnet: All-in-one control to text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 2023. 6, 7, 9, 19

[46] Dewei Zhou, You Li, Fan Ma, Xiaoting Zhang, and Yi Yang. Migc: Multi-instance generation controller for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6818–6828, 2024. 3