PGAMIE: A PIPELINE FOR GENERATING ACCURATE MULTI-INSTANCE IMAGE EDITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-guided image editing has advanced rapidly, enabling increasingly complex image modifications through natural-language instructions. Despite this progress, models often struggle to add or remove an exact number of object instances, particularly when instructions involve multiple object categories or edit operations. We present the first systematic evaluation of multi-instance editing and introduce PGAMIE, an automated pipeline for constructing verified multi-instance, multi-category image-editing data. The pipeline generates count-verified scenes and constructs verified addition and removal edits through instance segmentation, inpainting, and agreement-based count verification, supporting single-category, multi-category, and compound edits in scenes containing up to 44 object instances. Using this pipeline, we create MOMI, a 20k-record training dataset and a multi-category benchmark, together with two complementary benchmarks for out-of-distribution single-category and real-image evaluation. Our evaluation reveals that multi-instance editing remains challenging for current image editors, with performance degrading sharply as requested object counts and edit complexity increase. Fine-tuning Flux Kontext on MOMI achieves the highest edit accuracy on the MOMI benchmark at 16.3%, compared with 2.0% for the base model and 14.3% for Nanobanana Pro.

1 INTRODUCTION

Text-guided image editing transforms a source image according to a natural-language instruction (Brooks et al., 2023; Zhang et al., 2023; Black Forest Labs et al., 2025), and has become a standard tool for creating and adapting visual content. Many practical editing scenarios require precise control over object quantity. For example, a product shot may need to show exactly six bottles rather than the four that were photographed; a stock photograph may require removing three of its eight pedestrians and both parked scooters; and a catalog scene may require removing the two red chairs on the left and adding three stools in their place. Such edits require adding or removing a

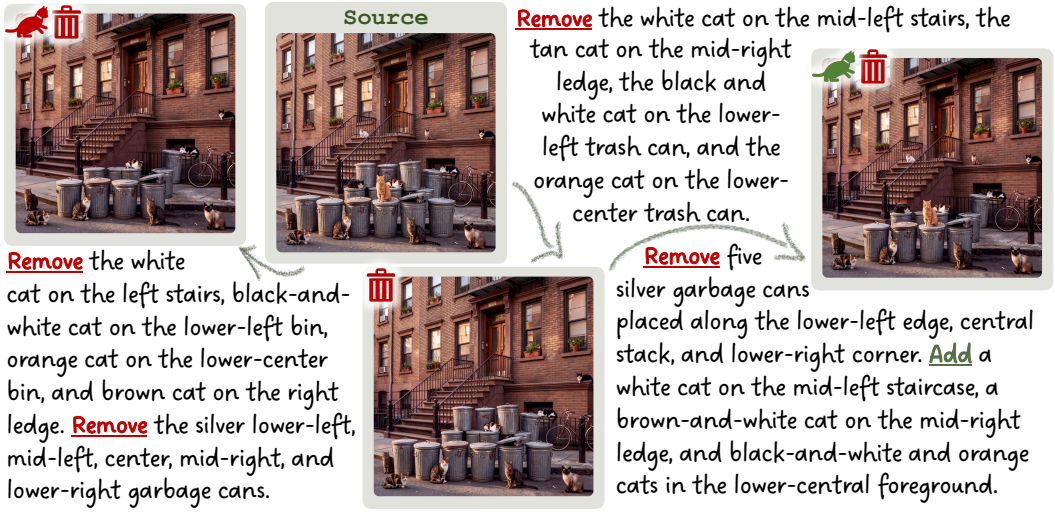


Figure 1: Verified multi-instance edits evaluated in this work, including two-category removal (left), single-category removal (top right), and compound remove+add editing (bottom right).

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.

specified number of instances of one or more object categories, sometimes both in a single instruction and often identified by color or position, while leaving the rest of the scene untouched (Fig. 1); a wrong count might render the result unusable. We refer to this task as multi-instance editing, distinct from applying separate edits to individual instances (Liu & Alaniz, 2026; Quattrini et al., 2026).

Current text-guided editors (Wu et al., 2025b; Black Forest Labs et al., 2025) handle this task poorly. Commercial and open-source models alike frequently fail to achieve the requested count, and the difficulty grows for instructions involving multiple object categories or edit operations.

Multi-instance edits appear in recent benchmarks only as a minor instruction type, with small requested counts and no analysis of how performance scales with count or scene complexity. We present the first evaluation that jointly covers large requested counts, dense scenes, multiple object categories, and compound add-and-remove operations, and analyze how each factor affects performance. Table 5 situates our evaluation scope among closely related benchmarks.

To enable this evaluation and to provide training data, we introduce PGAMIE: A Pipeline for Generating Accurate Multi-instance Image Edits, which constructs verified multi-instance edits automatically. Its design rests on two observations. First, image generators cannot reliably produce a requested number of objects (Cao et al., 2025), so we generate candidate scenes and keep only those on which two independent counters, SAM 3 instance segmentation (Carion et al., 2026) and Gemini visual question answering, agree, relabeling each scene with its verified count. Second, removing a chosen set of instances is easy with mask-guided inpainting, whereas adding an exact number is not; we therefore construct removal edits from SAM 3 masks, verify them the same way, and obtain additions by reversal (Wasserman et al., 2025), composing verified removals into multi-category and remove-and-add edits. The same counting mechanism underlies our automated evaluation, which agrees with human judgments on 96% of SOMI and MOMI outputs.

With this pipeline we build MOMI-Train (20,000 records across four edit types and six instruction types; Table 2) and three benchmarks: MOMI-Test (300 multi-category edits following the same taxonomy), SOMI (200 single-category edits with limited category overlap, distinct layouts, and a different generator from MOMI), and SOMI-Real (200 edits on Open Images photographs).

Our evaluation shows that multi-instance editing remains largely unsolved: no model exceeds 17% overall Edit Accuracy on MOMI-Test, accuracy falls roughly threefold as the number of spatial components grows from 1–3 to 4–6, and every model scores below 5% on combined remove-and-add instructions. Fine-tuning `Flux Kontext` on MOMI-Train raises Edit Accuracy from 2.0% to 16.3% on MOMI-Test, ahead of Nanobanana Pro (14.3%), and from 6.0% to 15.5% on SOMI and 6.5% to 13.5% on SOMI-Real. On the two out-of-distribution benchmarks it trails Nanobanana 2 and Pro (about 30%), but it is the most robust model as requested counts and scene density increase, and the strongest on removals and multi-object edits.

Contributions. Our contributions are: (1) the first evaluation of multi-instance image editing that jointly covers large requested counts, dense scenes, multiple object categories, and compound add-and-remove operations, across three benchmarks, together with a fine-grained analysis of how each factor affects performance; (2) PGAMIE, an automated pipeline that generates and verifies such edits, and MOMI-Train, the 20,000-record dataset it produces; and (3) a LoRA fine-tune of `Flux Kontext` on MOMI-Train that achieves the highest Edit Accuracy on MOMI-Test and more than doubles its base model on all three benchmarks.

2 Pipeline

2.1 Image generation

We generate high-quality, verified images containing up to 44 objects with diverse colors.

In the first stage (**Prompt Expansion and Generation**), object-count specifications such as "6 mushrooms" are expanded into image-generation prompts using Gemini 2.5 Pro. The initial prompt introduces diverse color attributes while encouraging distinct and clearly visible instances. MOMI candidates are generated using Nanobanana 2 (Google DeepMind, 2026), while SOMI uses Imagen 3 (Imagen-Team-Google, 2024). To generate three additional candidates for the same object-count specification, Gemini 2.5 Pro produces diverse prompts conditioned on the previous prompts, yielding four candidates total. The prompt templates used for the initial and diversity-based expansions are provided in Appendix .7.

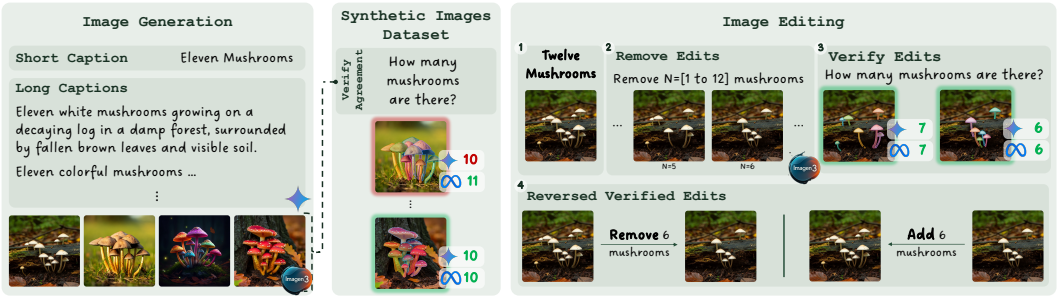


Figure 2: Overview of the PGAMIE pipeline generating SOMI benchmark edits. Given target categories and counts ("Eleven Mushrooms"), Gemini 2.5 Pro expands the input into a detailed caption, and Imagen 3 generates four candidates. Gemini-based VQA and SAM 3 verify their counts, updating metadata when needed ("Twelve white mushrooms..."). SAM 3 then extracts instance masks, Imagen 3 removes selected instances through inpainting for every requested count $i \in \{1, \ldots, N\}$, where N is the verified source count. The resulting edits are then verified by both models. Verified removal pairs are reversed to create additions, using the edited image as source and the original image as target. For MOMI multi-category images, we use the augmentation described in Section 2.2.

In the second stage (**Visual Question Answering (VQA) Generation**), we generate TIFA-style multiple-choice VQA queries (Hu et al., 2023) including counting questions (e.g., How many mushrooms are present?") and binary checks (e.g., Are there any mushrooms in the image?").

In the third stage (**Hybrid Instance-Segmentation and Vision-Language Verification**), object counts are evaluated using SAM 3 instance-mask counting and Gemini VQA. SAM 3 estimates the count as the number of target instance masks, suppressing segmentations of unrelated objects using negative mask focus areas. Gemini VQA provides a complementary semantic count, helping resolve ambiguities arising from part–whole fragmentation, overlapping objects, or small peripheral detections (Appendix .5). Candidates are accepted when SAM 3 and Gemini agree on the count, which differs from the requested count in 68.6% of accepted cases (e.g., 5 instead of 6 mushrooms).

2.2 Image Editing

Removal Generation and Quality Control We extract instance-level segmentation masks for target objects using SAM 3. For each verified source image containing N target instances, we construct removal edits for counts $i \in \{1, \ldots, N\}$, producing an inpainting mask composed of the selected i instance masks. For MOMI train, we exclude edits that remove all target instances ($i = N$) to discourage models from learning to remove the entire target category regardless of the requested count; see Appendix .2 for further analysis of this bias. To reduce residual boundary artifacts, such as colored outlines of removed objects, edit masks are adaptively dilated according to object size before inpainting. Object removal is performed using Imagen 3. If SAM 3 detects residual target remnants within the edit region, a second-pass removal is performed. To preserve background consistency, pixels outside the expanded mask are restored from the source image. Edits undergo two quality checks: we discard those whose edit mask covers less than 0.1% of the image to exclude excessively small edit regions, or whose CLIP similarity to the source is ≤ 0.75, indicating substantial unintended changes. Overall, 65.5% of executed edits satisfy these quality criteria.

Verification and Addition Reversal We retain only removal edits whose post-edit count matches the expected count by both SAM 3 and Gemini, resulting in a 56% pass rate. Because text-guided addition inpainting often produces incorrect instance counts or visual artifacts, we construct additions by reversing the verified removal edits, following prior work (Wasserman et al., 2025): the edited image becomes the source, the original image becomes the target, and the instruction is inverted.

Multi-Category and Compound Edit Augmentation For multi-category images, we construct multi-category removals by applying a second verified removal to a different category. For example, after removing two bowls, we additionally remove three pillows. The original source image and the second-

Table 1: Representative examples of four edit types.

Type	Example
Addition	Add three pillows.
Removal	Remove two bowls.
Multi-Obj.	Remove two bowls and three pillows.
Same-Act.	
Remove+Add	Remove two bowls and add three pillows.

stage edited image form the final edit pair,
combining the two removal instructions into a single instruction (e.g., "remove two bowls and three pillows"). We also construct Remove+Add edits from two verified removals from the same source image: the image with category A removed serves as the source and the image with category B removed as the target, yielding "add two bowls and remove three pillows." Reversing the multi-category and Remove+Add pairs produces multi-category additions and the opposite Remove+Add direction. Together, these constructions yield four edit types: *Addition*, *Removal*, *Multi-Object Same-Action*, and *Remove+Add* (Table 1).

Table 2: The six instruction types illustrated with representative instructions.

Instruction	Example
Basic	Remove two bowls and add three pillows.
Color	Remove two pink bowls and add three blue pillows.
Spatial	Remove two bowls in the top left and add three pillows in the bottom right.
Color-Spatial	Remove two pink bowls in the top left and add three blue pillows in the bottom right.
Detailed-Spatial	Remove two bowls in the top left corner and add three pillows on the bottom right couch.
Color-Detailed-Spatial	Remove two pink bowls in the top left corner and add three blue pillows on the bottom right couch.

Instruction Types and Attribute Enrichment For each verified edit pair, we generate six instruction variants with varying attribute detail: *Basic*, *Color*, *Spatial*, *Color-Spatial*, *Detailed-Spatial*, and *Color-Detailed-Spatial* (Table 2). *Basic* specifies the edit operation, object category, and count (e.g., "Remove two bowls and add three pillows"). *Color* adds instance-level color attributes from SAM 3 segmentations, *Spatial* adds coarse regional descriptors, *Detailed-Spatial* adds fine-grained scene-relative spatial descriptors, and *Color-Spatial* and *Color-Detailed-Spatial* add color attributes to the corresponding spatial variants. Spatial descriptors are extracted using a single Gemini 3.1 Pro prompt, which receives the source image for removals and the target image for additions, overlaid with bounding boxes from the corresponding SAM 3 instance masks (Appendix .7).

3 DATASET

Using our pipeline, we construct a large-scale training dataset and three complementary evaluation benchmarks for evaluating multi-instance image editing across diverse settings.

Table 3: Summary of the training and evaluation datasets. Requested count denotes the number of object instances modified per instruction.

Dataset	Samples	Image Domain	Scene Setting	Categories	Edit Type	Mean Req. Count	Max Req. Count
MOMI Train	20,000	Synthetic	Multi-category	52	4	12.49	30
MOMI Test	300	Synthetic	Multi-category	42	4	12.37	28
SOMI Test	200	Synthetic OOD	Single-category	100	2	5.50	10
SOMI-Real Test	200	Real	Single-category	16	2	6.78	15

MOMI Training Dataset The **Multi-Object Multi-Instance (MOMI)** training set comprises 20,000 instruction-conditioned training records derived from 13,351 unique source–target image pairs (Appendix .2). Each record consists of a source image, an edited target image, and a text instruction specifying multi-instance modifications within multi-category scenes, targeting one or two object categories (e.g., "Remove two bananas and add three mangoes"). We use this dataset to fine-tune the `Flux Kontext` base model.

Evaluation Benchmarks **MOMI** evaluates complex multi-category editing across six instruction types and four edit types; **SOMI** evaluates out-of-distribution single-category editing using different object categories, scene layouts, and a different image generation model from MOMI; and **SOMI-Real** evaluates zero-shot generalization to in-the-wild photographs from Open Images.

Dataset Statistics Table 3 summarizes the training and evaluation splits. MOMI Train/Test comprise *Multi-Object Same-Action* (32.3%/36.3%), *Remove+Add* (26.2%/21.0%), *Add* (20.8%/21.0%), and *Remove* (20.7%/21.7%) edits. MOMI Train and MOMI Test are also approximately balanced across the six instruction types ($\sim$ 16% each), whereas SOMI and SOMI-Real

contain Basic instructions only. Instructions target one object category in approximately 42% of MOMI records and two categories in approximately 58%. During training, we observed a bias toward removing all target instances. To mitigate this, we exclude full-removal edits and balance the edited-object fraction (Appendix .2). Additional dataset statistics are provided in Appendix .2, with qualitative examples in Appendix .3.

Image Lineage and Leakage Prevention To prevent train–test leakage, we partition the data at the *root prompt and image lineage level* before constructing reversed addition pairs, compound edits, or multi-category same-action edits. All edits derived from the same original generated scene or base prompt are assigned exclusively to a single split. This prevents overlap of endpoint images, background scenes, and prompt lineages between the training and evaluation splits.

4 EXPERIMENTS

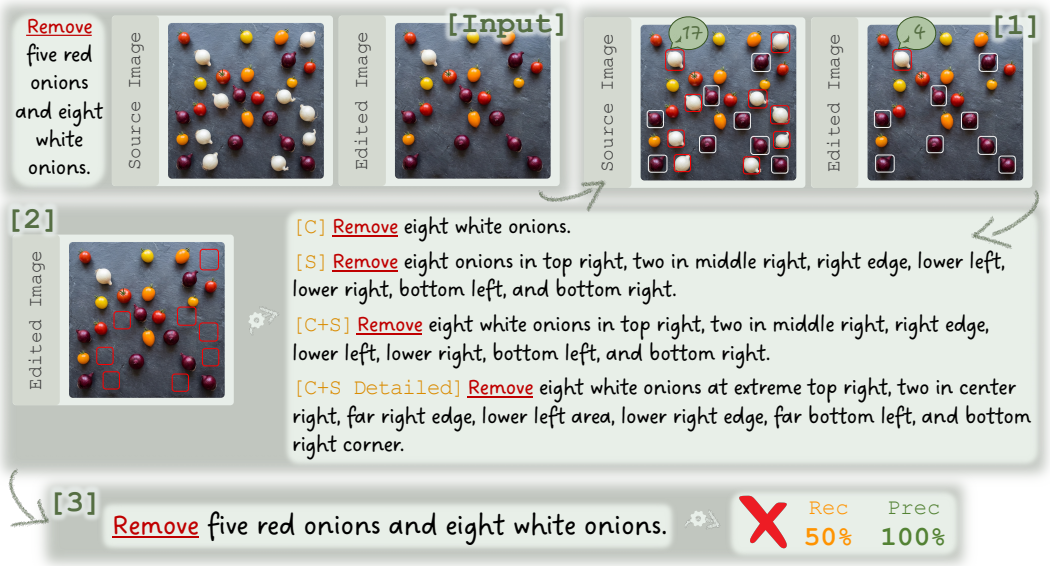


Figure 3: Overview of our automated evaluation pipeline. [1] SAM 3 segments and counts relevant objects in the source and edited images to infer edit actions and quantities. [2] Color and spatial attributes are added to match the ground-truth instruction type. [3] Gemini 3.1 Pro compares inferred and ground-truth instruction components to compute the evaluation metrics. In the example, Edit Accuracy fails, while color recall and precision are 50% and 100%, respectively.

We systematically evaluate multi-instance editing across the MOMI, SOMI, and SOMI-Real benchmarks. We compare a `Flux Kontext` model fine-tuned on the MOMI training set using LoRA, denoted PGAMIE Model, against nine recent image editors: BAGEL (Deng et al., 2025), Flux Klein (Black Forest Labs, 2026), Flux Kontext (Black Forest Labs et al., 2025), Nanobanana (Google, 2025), Nanobanana Pro (Google DeepMind, 2025), Nanobanana 2 (Google DeepMind, 2026), OmniGen2 (Wu et al., 2025b), Qwen Image Edit (Wu et al., 2025a), and Step1X (Liu et al., 2025). Full training and implementation details are provided in Appendix .8.

Evaluation Procedure Our evaluation consists of three stages, illustrated in Fig. 3.

First, we infer a basic instruction describing the edit actually performed by the model from the source and edited images. We use SAM 3 to segment the relevant object categories in both images and compare their instance counts to infer the performed edit actions and quantities, producing an instruction such as *"remove three cups and add one bottle."*

Second, we enrich the inferred basic instruction to match the structure of the ground-truth instruction type. When the ground-truth instruction contains color or spatial attributes, Gemini 3.1 Pro infers the corresponding attributes from the source and edited images and adds them to the inferred instruction.

Third, Gemini 3.1 Pro compares the inferred and ground-truth instructions using a structured component-based evaluation. It decomposes both instructions into edit, color, and spatial com-

ponents and evaluates whether each requested component matches the corresponding inferred component, while identifying missing and extra components.

Human Validation of the Automated Evaluation Pipeline. We validate the pipeline on both SOMI and MOMI. On SOMI, all 200 samples are evaluated for BAGEL, Qwen Image Edit, and Flux Kontext by three independent annotators, with majority vote used as the human reference. The pipeline achieves 96.0–96.5% agreement with human judgments ($\kappa = 0.615$–0.792), with model-level Edit Accuracy differing by at most 2.0 percentage points. On MOMI, we conduct targeted expert validation on 100 stratified MOMI-Test examples for each of PGAMIE and Nanobanana 2. The automated evaluator achieves 96.0% agreement on Edit Accuracy ($\kappa = 0.839$) and 87.3% agreement on color correctness. Full validation details are provided in Appendix .4. Additional validation includes lexical-metric stress tests (Appendix .7) and a comparison of SAM 3 against YOLO-World for object grounding (Appendix .4).

4.1 Evaluation Metrics

Our primary instruction-following metric is **Edit Accuracy**, which requires an exact match in edit type, object, and count while excluding color and spatial attributes. We additionally report **Color Component Metrics**, measuring precision and recall over color-object associations with soft matching for semantically equivalent colors, and **Editing Failure Modes**, including no-edit failures, incomplete add-and-remove edits, opposite-action errors, and unrequested additions or removals. We exclude color and spatial constraints as most models fail when they are included (Appendix .6).

4.2 Quantitative Results: Multi-Object Multi-Instance Benchmark

Table 4: Edit Accuracy across benchmarks, measured by exact edit type, object, and count match. Nano abbreviates the corresponding Nanobanana models. On MOMI, PGAMIE achieves the highest overall accuracy (16.3%) and leads on four of six instruction types; Nano Pro leads on the two spatial-only types. Nano 2 leads on SOMI and SOMI-Real.

	BAGEL	Flux Klein	Flux Kontext	PGAMIE	Nano Pro	Nano	Nano 2	Omni Gen2	Qwen Image Edit	Step1X
Multi-Object Multi-Instance (MOMI)										
Basic	8.3%	2.1%	2.1%	**16.7%**	12.5%	4.2%	6.3%	2.1%	2.1%	0.0%
Color	0.0%	1.8%	0.0%	**12.5%**	8.9%	1.8%	10.7%	0.0%	1.8%	0.0%
Spatial	0.0%	5.9%	0.0%	11.8%	**13.7%**	3.9%	9.8%	0.0%	2.0%	0.0%
Color-Spatial	6.3%	6.3%	2.1%	**20.8%**	14.6%	8.3%	14.6%	10.6%	8.3%	0.0%
Detailed-Spatial	0.0%	12.0%	2.0%	10.0%	**14.0%**	0.0%	12.0%	0.0%	2.0%	1.9%
Color-Detailed-Spatial	8.5%	6.4%	6.4%	**27.7%**	23.4%	10.6%	23.4%	2.1%	4.0%	2.1%
All	3.7%	5.7%	2.0%	**16.3%**	14.3%	4.7%	12.7%	2.3%	3.3%	0.7%
Single-Object Multi-Instance (SOMI)										
Basic	5.5%	7.5%	6.0%	15.5%	28.5%	14.5%	**29.5%**	3.0%	10.0%	2.5%
Single-Object Multi-Instance Real (SOMI-Real)										
Basic	6.5%	8.5%	6.5%	13.5%	28.9%	14.0%	**30.0%**	5.0%	8.5%	5.5%

Performance on the multi-category MOMI benchmark reveals that current models struggle with instructions involving multiple objects. PGAMIE achieves the highest overall accuracy at 16.3% and highest accuracy on four of the six evaluated instruction types. Nanobanana Pro ranks second with 14.3% overall accuracy and achieves the best performance on the remaining two instruction types. Other top-tier models perform poorly even on the simplest *Basic instructions*: Step1X (Liu et al., 2025) achieves 0%, Flux Kontext 2.1%, and OmniGen2 2.1% (Table 4).

4.3 Single-Category Benchmarks

On SOMI, Nanobanana 2 achieves the highest accuracy (29.5%), followed by Nanobanana Pro (28.5%), while PGAMIE achieves 15.5%. Performance is generally higher than on the multi-category MOMI benchmark, where the best model reaches 16.3%. This gap is consistent with SOMI's lower scene density, averaging 9 objects per image versus 25 for MOMI, and with the performance degradation observed as scene object count increases in Section 4.4.

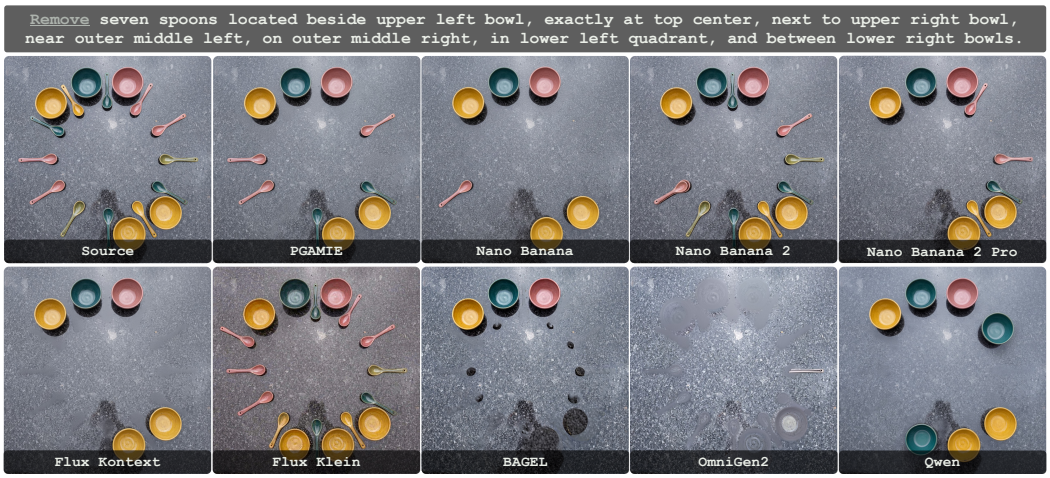


Figure 4: Qualitative comparison on a seven-removal spatial instruction. PGAMIE is the only successful model, while others show partial, excessive, or remove-all edits and unintended changes.

Performance also differs between addition and removal. On SOMI, for example, Qwen drops from 19.0% on additions to 1.0% on removals, while BAGEL drops from 10.0% to 1.0%. In contrast, PGAMIE remains balanced across both operations: 16.0% vs. 15.0% on SOMI and 13.0% vs. 14.0% on SOMI-Real (Appendix, Figures 31 and 32).

To evaluate generalization beyond training images, we test all models on the SOMI-Real benchmark. As shown in Table 4, PGAMIE achieves 13.5% Edit Accuracy on real images, compared with 15.5% on SOMI. While PGAMIE substantially improves over open-source editors such as Flux Kontext (6.5%) and OmniGen2 (5.0%), it trails commercial baselines such as Nanobanana Pro (28.9%) and Nanobanana 2 (30.0%). This performance gap illustrates partial zero-shot transfer from training supervision while highlighting that bridging the domain gap remains an important direction for in-the-wild multi-instance editing.

4.4 Fine-Grained Complexity Analysis

Increased scene object count leads to performance decline. Across models, Edit Accuracy decreases as the number of objects in the image increases (Appendix, Figures 46, 47). PGAMIE shows the smallest decline from scenes with fewer than 17 objects (23.6%) to scenes with 26–32 objects (17.4%), whereas Nanobanana Pro drops from 27.8% to 8.7%. However, in scenes containing at least 33 objects, PGAMIE falls sharply to 2.8%, while Nanobanana Pro maintains its performance (8.3%). Overall, PGAMIE is more robust to moderate increases in scene object count, but this advantage diminishes in extremely crowded scenes.

Edit Accuracy declines sharply as the number of requested objects increases. Most models show a threefold accuracy drop for instructions involving 6–12 objects (Figure 49). All models reach 0.0% in the 19–24 objects range except PGAMIE (4.4%), Nanobanana Pro (2.2%), and Nanobanana 2 (2.2%). PGAMIE alone succeeds on edits involving at least 25 objects.

Edit Accuracy declines with increasing spatial and color complexity. Edit Accuracy declines sharply as spatial and color complexity increases (Appendix, Figures 50 and 51). From 1–3 to 4–6 spatial components, accuracy drops roughly threefold or more for most models; from 1–2 to 3–4 color components, it falls by about half or more. PGAMIE performs best in every spatial-complexity bin except 1–3 components, and leads for 1–2 color components while tying for best at 5 or more. Figure 4 illustrates this difficulty on a seven-removal detailed-spatial instruction, where PGAMIE is the only model to complete the requested edit.

Color and spatial guidance are most effective when combined. Across nearly all models, color-spatial instructions achieve higher edit accuracy than color and spatial instructions (Table 4). For example, PGAMIE improves from 12.5% with color-only instructions and 11.8% with spatial-only instructions to 20.8% when both cues are combined. Similar trend hold across models. Adding detailed spatial guidance to color instructions further improves performance for most models, with PGAMIE reaching 27.7%. These results suggest that color and spatial cues provide complementary information for identifying the intended objects and executing the requested edit type and count.

Color attributes are more often omitted than incorrectly applied. Models achieve high color precision (80.0%) alongside moderate recall (53.6%). This gap indicates that when models attempt a color edit, they generally apply the correct color to the target object, whereas failures stem primarily from omitting requested color attributes rather than applying wrong colors. PGAMIE achieves a strong balance between color precision (78.6%) and recall (68.8%), maintaining high color accuracy while successfully completing color modifications across a larger proportion of requested objects.

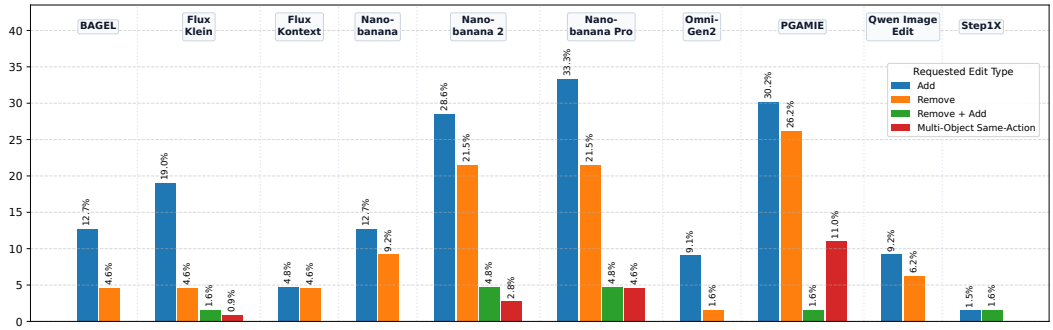


Figure 5: Edit Accuracy by edit type on MOMI. Models perform better on isolated addition and removal than on *Remove+Add*. PGAMIE leads on removal and *Multi-Object Same-Action*; Nano Pro leads on addition and ties with Nano 2 on *Remove+Add*.

4.5 Edit-Type Analysis

Models generally perform better on additions than removals across datasets. Across the three benchmarks, Edit Accuracy is generally higher for additions than removals. On MOMI, Nanobanana Pro drops from 33.3% to 21.5%. PGAMIE is more balanced at 30.2% and 26.2%, respectively, achieving the highest removal accuracy (Figure 5).

Combined addition and removal edits remain a major bottleneck across all models. Performance drops sharply when additions and removals are combined in one instruction (Figure 5). All models score below 5%, with Nanobanana Pro and Nanobanana 2 reaching 4.8%, while PGAMIE, Flux Klein, and Step1X achieve 1.6%, and remaining models score 0.0%. Despite achieving 11.0% on multi-object same-action edits, PGAMIE drops to 1.6% on combined addition and removals.

PGAMIE is the most robust model on multi-object same-action edits. When adding or removing two object categories, PGAMIE achieves the highest accuracy at 11.0% (Figure 5), more than twice Nanobanana Pro (4.6%) and nearly four times Nanobanana 2 (2.8%). All other models except Flux Klein (0.9%) score 0.0%, reflecting the difficulty of multi-category editing.

4.6 Error Analysis

No-edit failures occur primarily on additions, while PGAMIE rarely fails to perform an edit. Addition requests produce the highest no-edit rate for nine of the ten evaluated models, ranging from 3.2% to 49.2% (Appendix .6). PGAMIE (0.7%), Nanobanana Pro (0.7%), and Nanobanana 2 (1.3%) rarely fail to perform an edit across all edit types. In contrast, the overall no-edit rate reaches 19.7% for OmniGen2 and 30.3% for Flux Kontext (Appendix, Figure 26).

PGAMIE most often performs both requested actions in combined remove-and-add edits. PGAMIE performs only one of the two requested actions in just 9.5% of cases—for example, adding the requested object without removing the other one (Figure 30). This occurs in 15.9% of cases for Nanobanana 2, 17.5% for Nanobanana Pro, 43.5% for Qwen Image Edit, and 87.1% for Step1X.

PGAMIE shows the least add/remove confusion and the lowest Extra Unrequested Component Rate among open-source models. PGAMIE has the lowest opposite-action error rate at 1.3%, versus 23.7% for OmniGen2 and 25.3% for Step1X (Figure 29). Its Extra Unrequested Component Rate is 11.0%, below all open-source models and close to Nanobanana 2 (9.7%) and Nanobanana Pro (5.3%), while Step1X and OmniGen2 reach 34.3% and 37.7% (Figure 27).

5 Related Work

Instruction-guided image editing. Recent advances in text-guided image editing enable diverse local and global modifications via natural-language instructions (Liu et al., 2025; Wu et al., 2025a;

Table 5: Comparison of relevant image editing benchmarks. ✓* indicates that the property is present in a limited subset of benchmark examples ($< 3\%$ of samples) rather than being a systematic evaluation axis. All cells are verified directly from released benchmark data (see Appendix .1 for details).

Benchmark	Explicit count	Req. count > 5	Dense scenes (≥ 20)	Multi-category	Add+ Rem.	Count+ col./ spat.	Count + col.+ spat.	Fine/ Error Analysis	Training superv.	Synth & Real
QL-Dataset (Tan et al., 2026)	✓	–	–	–	–	–	–	–	✓	✓
EdiVal-Bench (Chen et al., 2026)	✓	–	–	–	–	–	–	–	–	✓
DLEBench (Hong et al., 2026)	✓	–	–	–	–	✓*	✓*	–	–	–
MIRA-Bench (Liu & Alaniz, 2026)	–	–	–	✓	✓	–	–	–	–	–
MICE-Bench (Quattrini et al., 2026)	–	–	✓	✓	–	–	–	–	–	–
HarmonyBench (Shen et al., 2025)	–	–	–	✓	–	–	–	–	✓	✓
UniEditBench (Jiang et al., 2026)	✓	–	–	–	–	✓*	–	–	–	✓
PGAMIE *(Ours)*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Black Forest Labs et al., 2025; Wu et al., 2025b; Brooks et al., 2023; Zhao et al., 2024). Other approaches incorporate mask-guided editing and automatically derived edit masks, to constrain modifications to target regions (Nguyen et al., 2025; He et al., 2026).

Count-aware generation and editing. Exact object counting has been extensively studied in text-to-image generation, with benchmarks such as GenEval, T2I-CompBench++, and T2ICountBench demonstrating persistent counting failures, and methods such as CountGen explicitly improving count control during generation (Ghosh et al., 2023; Huang et al., 2025; Cao et al., 2025; Binyamin et al., 2025). In image editing, IMAGHarmony preserves object cardinality and layout while modifying semantics (Shen et al., 2025), while QL-Adapter incorporates explicit object-layout information for multi-object editing (Tan et al., 2026). In contrast, we evaluate explicit count-changing instructions involving up to 30 instances, multiple object categories, and compound add/remove operations across synthetic and real images. Table 5 summarizes these distinctions.

Automated evaluation. Several image-editing benchmarks employ automated evaluation methods to assess semantic, structural, and preservation criteria (Basu et al., 2023; Ju et al., 2023; Ma et al., 2024; Qian et al., 2025; Pan et al., 2025), with some further validating these metrics against human judgments (Basu et al., 2023; Ma et al., 2024; Qian et al., 2025). Our evaluation similarly uses an automated procedure that decomposes inferred and ground-truth edit instructions into edit, color, and spatial components, enabling fine-grained analysis of editing correctness.

6 Conclusion and Future Work

We presented the first systematic evaluation of multi-instance image editing, showing that current editors struggle as object counts and edit complexity increase. We introduced PGAMIE, an automated pipeline for generating and verifying multi-instance, multi-category edits, and used it to construct the 20k-record MOMI training set and the MOMI, SOMI, and SOMI-Real benchmarks. We further demonstrate the value of MOMI as a training resource: fine-tuning `Flux Kontext` on MOMI increases Edit Accuracy from 2.0% to 16.3%, surpassing Nanobanana Pro at 14.3%.

Future Work Future work can develop methods that better handle high object counts, multiple categories, and compound edits. Another direction is to study the sequential dynamics of multi-component instructions, including how component order affects execution accuracy and how earlier failures influence later components and overall completion. Extending multi-instance editing to video and 3D scenes is also a promising direction.

7 Limitations

Our approach has several limitations. First, the MOMI training dataset is primarily constructed from generated base images and may therefore not fully capture the diversity, occlusions, object interactions, and visual complexity of real-world scenes. Although SOMI-Real evaluates generalization to real images, it is limited to single-category Basic instructions and therefore does not assess real-world performance on multi-category or compound edits. Second, our current training and evaluation setting focuses primarily on object addition and removal and does not extensively cover other editing operations, such as object replacement, geometric transformations, or fine-grained appearance changes.

AI USE STATEMENT

Generative AI tools were used as part of the research methodology described in this paper. Gemini 2.5 Pro was used for prompt expansion and visual question answering, Gemini 3.1 Pro for spatial-attribute generation and structured instruction evaluation, Nanobanana 2 and Imagen 3 for synthetic image generation, and Imagen 3 for inpainting-based object removal. These uses are described in detail in the corresponding methodology sections. ChatGPT was additionally used to assist with language editing and polishing of the manuscript. All AI-assisted content was reviewed by the authors, who take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies multi-instance image editing using primarily synthetically generated images, together with a real-image benchmark constructed from Open Images V7. Human participation was limited to validation of the automated evaluation pipeline using annotators recruited through Amazon Mechanical Turk (MTurk), who assessed whether model-generated image edits satisfied the provided editing instructions. No personal or sensitive information about annotators was collected or analyzed, and the human-annotation study did not require institutional ethics review. The proposed benchmarks focus on generic object-level addition and removal rather than identity- or person-specific manipulation. Synthetic images were generated using Google generative AI services under their applicable terms, which do not claim ownership over original generated content or new intellectual property created in generated outputs. For SOMI-Real, Open Images V7 lists its images under CC BY 2.0 and its annotations under CC BY 4.0; we preserve the corresponding licensing and attribution information in the released benchmark. As with general-purpose image-editing systems, the methods evaluated in this work may potentially be used in unintended ways; our work is intended to improve the measurement and understanding of such systems rather than enable harmful applications.

REPRODUCIBILITY STATEMENT

We provide detailed information to support reproducibility throughout the paper and appendix. The dataset construction procedure, verification criteria, and benchmark composition are described in Sections 2 and 3 and Appendix .2. Full evaluation methodology and prompt specifications are provided in Appendix .7. Appendix .4 details the human-validation protocols, while Appendix .8 reports the fine-tuning configuration and hyperparameters. We additionally document dataset partitioning and image-lineage constraints used to prevent train–test leakage in Appendix .9. We plan to release the code and datasets upon publication, with third-party content distributed or referenced in accordance with its applicable source licenses.

REFERENCES

Samyadeep Basu, Mehrdad Saberi, Shweta Bhardwaj, Atoosa Malemir Chegini, Daniela Massiceti, Maziar Sanjabi, Shell Xu Hu, and Soheil Feizi. Editval: Benchmarking diffusion based text-guided image editing methods. *arXiv preprint arXiv:2310.02426*, 2023.

Lital Binyamin, Yoad Tewel, Hilit Segev, Eran Hirsch, Royi Rassin, and Gal Chechik. Make it count: Text-to-image generation with an accurate number of objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13242–13251, June 2025.

Black Forest Labs. FLUX.2 [klein]. https://huggingface.co/black-forest-labs/FLUX.2-klein-4B, 2026. Model card, released January 2026.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. URL https://arxiv.org/abs/2506.15742.

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 (CVPR)*, pp. 18392–18402, June 2023.

Yuefan Cao, Xuyang Guo, Jiayan Huo, Yingyu Liang, Zhenmei Shi, Zhao Song, Jiahao Zhang, and Zhen Zhuang. Text-to-image diffusion models cannot count, and prompt refinement cannot help. *arXiv preprint arXiv:2503.06884*, 2025.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts, 2026. URL https://arxiv.org/abs/2511.16719.

Tianyu Chen, Yasi Zhang, Zhi Zhang, Peiyu Yu, Shu Wang, Zhendong Wang, Kevin Lin, Xiaofei Wang, Zhengyuan Yang, Linjie Li, Chung-Ching Lin, Jianwen Xie, Oscar Leong, Lijuan Wang, Ying Nian Wu, and Mingyuan Zhou. Edival-agent: An object-centric framework for automated, fine-grained evaluation of multi-turn editing. In *International Conference on Learning Representations*, 2026.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Guang Shi, and Haoqi Fan. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

Dhruba Ghosh, Hanna Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *arXiv preprint arXiv:2310.11513*, 2023.

Google. Introducing Gemini 2.5 Flash Image (Nano Banana), our state-of-the-art image model. https://developers.googleblog.com/en/introducing-gemini-2-5-flash-image/, 2025.

Google DeepMind. Nano Banana Pro: Gemini 3 Pro Image. https://blog.google/innovation-and-ai/products/nano-banana-pro/, 2025.

Google DeepMind. Nano Banana 2: Gemini 3.1 Flash Image. https://blog.google/innovation-and-ai/technology/ai/nano-banana-2/, 2026.

Jingxuan He, Xiyu Wang, Mengyu Zheng, Xiangyu Zeng, Yunke Wang, and Chang Xu. Rethinking where to edit: Task-aware localization for instruction-based image editing. *arXiv preprint arXiv:2604.20258*, 2026.

Shibo Hong, Boxian Ai, Jun Kuang, Wei Wang, FengJiao Chen, Zhongyuan Peng, Chenhao Huang, and Yixin Cao. Dlebench: Evaluating small-scale object editing ability for instruction-based image editing model. In *Proceedings of the International Conference on Machine Learning*, 2026.

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A Smith. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *2023 ieee/cvf international conference on computer vision (iccv)*, pp. 20349–20360. IEEE, 2023.

Kaiyi Huang, Chengqi Duan, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench++: An enhanced and comprehensive benchmark for compositional text-to-image generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(5):3563–3579, 2025. doi: 10.1109/TPAMI.2025.3531907.

Imagen-Team-Google. Imagen 3. *arXiv preprint arXiv:2408.07009*, 2024.

Lifan Jiang, Tianrun Wu, Yuhang Pei, Chenyang Wang, Boxi Wu, and Deng Cai. Unieditbench: A unified and cost-effective benchmark for image and video editing via distilled mllms. *arXiv preprint arXiv:2604.15871*, 2026.

Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Direct inversion: Boosting diffusion-based editing with 3 lines of code. *arXiv preprint arXiv:2310.01506*, 2023.

Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, Guopeng Li, Yuang Peng, Quan Sun, Jingwei Wu, Yan Cai, Zheng Ge, Ranchen Ming, Lei Xia, Xianfang Zeng, Yibo Zhu, Binxing Jiao, Xiangyu Zhang, Gang Yu, and Daxin Jiang. Step1x-edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025.

Ziqian Liu and Stephan Alaniz. Mirage: Benchmarking and aligning multi-instance image editing. *arXiv preprint arXiv:2604.05180*, 2026.

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

Trong-Tung Nguyen, Quang Nguyen, Khoi Nguyen, Anh Tran, and Cuong Pham. Swiftedit: Lightning fast text-guided image editing via one-step diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21492–21501, 2025.

Yulin Pan, Xiangteng He, Chaojie Mao, Zhen Han, Zeyinzi Jiang, Jingfeng Zhang, and Yu Liu. Icebench: A unified and comprehensive benchmark for image creating and editing. *arXiv preprint arXiv:2503.14482*, 2025.

Yusu Qian, Jiasen Lu, Tsu-Jui Fu, Xinze Wang, Chen Chen, Yinfei Yang, Wenze Hu, and Zhe Gan. Gie-bench: Towards grounded evaluation for text-guided image editing. *arXiv preprint arXiv:2505.11493*, 2025.

Fabio Quattrini, Carmine Zaccagnino, Enis Simsar, Marta Tintoré Gazulla, Rita Cucchiara, Alessio Tonioni, and Silvia Cascianelli. Editing everything everywhere all at once. In *Proceedings of the European Conference on Computer Vision*, 2026.

Fei Shen, Xiaoyu Du, Yutong Gao, Jian Yu, Yushe Cao, Xing Lei, and Jinhui Tang. Imagharmony: Controllable image editing with consistent object quantity and layout. *arXiv preprint arXiv:2506.01949*, 2025.

Jiaqi Tan, Fangyu Li, and Yang Liu. Instruction guided multi object image editing with quantity and layout consistency. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8912–8916. IEEE, 2026.

Navve Wasserman, Noam Rotstein, Roy Ganz, and Ron Kimmel. Paint by inpaint: Learning to add image objects by removing them first. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18313–18324, June 2025.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Shengming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025a.

Chenyuan Wu, Pengfei Zheng, Ruiran Yan, Shitao Xiao, Xin Luo, Yueze Wang, Wanli Li, Xiyan Jiang, Yexin Liu, Junjie Zhou, Ze Liu, Ziyi Xia, Chaofan Li, Haoge Deng, Jiahao Wang, Kun Luo, Bo Zhang, Defu Lian, Xinlong Wang, Zhongyuan Wang, Tiejun Huang, and Zheng Liu. Omnigen2: Exploration to advanced multimodal generation, 2025b. URL https://arxiv.org/abs/2506.18871.

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. Magicbrush: A manually annotated dataset for instruction-guided image editing. In *Advances in Neural Information Processing Systems*, volume 36, 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. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

.1 Comparison with Prior Multi-Instance and Quantity-Aware Benchmarks

Prior work has studied quantity-aware editing, quantity-preserving editing, and concurrent editing of multiple object instances under complementary settings. To our knowledge, MOMI is the first benchmark to jointly evaluate instructions requesting edits to more than ten object instances, multi-category instructions, and compound add-and-remove operations. Specifically, MOMI Test contains single instructions requesting modifications to as many as 28 object instances, while MOMI Train contains instructions modifying up to 30. Table 6 compares our evaluation setting with closely related image-editing benchmarks and evaluation suites.

Table 6: Comparison with prior benchmarks and evaluation suites related to quantity-aware and multi-instance image editing across 10 verified properties. A checkmark (✓) indicates that the corresponding property is systematically represented and evaluated; an asterisked checkmark (✓*) indicates that the property is present in a limited subset of examples ($< 3\%$ of samples) rather than forming a systematic evaluation axis; a dash (–) indicates that it is absent.

Benchmark	Explicit count	Req. count > 5	Dense scenes (≥ 20)	Multi-category	Add+ Rem.	Count+ col./ spat.	Count + col.+ spat.	Fine/ Error Analysis	Training superv.	Synth & Real
QL-Dataset (Tan et al., 2026)	✓	–	–	–	–	–	–	–	✓	✓
EdiVal-Bench (Chen et al., 2026)	✓	–	–	–	–	–	–	–	–	✓
DLEBench (Hong et al., 2026)	✓	–	–	–	–	✓*	✓*	–	–	–
MIRA-Bench (Liu & Alaniz, 2026)	–	–	–	✓	✓	–	–	–	–	–
MICE-Bench (Quattrini et al., 2026)	–	–	✓	✓	–	–	–	–	–	–
HarmonyBench (Shen et al., 2025)	–	–	–	✓	–	–	–	–	✓	✓
UniEditBench (Jiang et al., 2026)	✓	–	–	–	–	✓*	–	–	–	✓
PGAMIE *(Ours)*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Property definitions and verification criteria. The properties evaluated in Table 6 are defined as follows:

- **Explicit count:** Natural-language instructions explicitly requesting addition or removal of a specified numerical quantity of objects.
- **Req. count > 5:** Contains test instructions requesting modifications exceeding 5 instances in a single prompt (e.g., MOMI evaluates up to 28 requested instances).
- **Dense scenes (≥ 20):** Evaluates images containing 20 or more simultaneous object instances.
- **Multi-category:** Jointly modifies multiple distinct object categories within a single edit instruction.
- **Add+Rem.:** Executes simultaneous addition and removal of objects within a single prompt.
- **Count+col./spat.:** Couples count modifications with color appearance or spatial localization descriptors. An asterisk (✓*) denotes incidental phrasing appearing in a limited subset ($< 3\%$ of samples) rather than forming a systematic evaluation split.
- **Count + col.+spat.:** Jointly couples count modifications with both color and spatial attributes.
- **Fine/Error Analysis:** Evaluates continuous multi-axis complexity stratification (≥ 2 varied axes) and decomposed operational error modes (isolating opposite-action, no-edit omission, and unrequested edit rates).
- **Training superv.:** Provides a dedicated training dataset for supervised instruction fine-tuning.

- **Synth & Real:** Evaluates performance across both synthetically generated scenes and real-world photographic imagery.

.2 Comprehensive Dataset Statistics

This section reports the descriptive statistics of the MOMI training set and the three evaluation benchmarks. All values are computed directly from the structured ground-truth metadata. We use no separate validation split.

Dataset Composition

MOMI Train contains 20,000 instruction-conditioned records, and the evaluation suite contains 700 samples. MOMI Train, MOMI Test, and SOMI Test use synthetic images, whereas SOMI-Real contains real photographs from Open Images. Across all four splits, 20,500 records (99.03%) are synthetic and 200 (0.97%) are real. Considering only the evaluation suite, 500 samples are synthetic and 200 are real.

Table 7: Composition of the training and evaluation splits.

Split	Records	Synthetic	Real	Categories
MOMI Train	20,000	20,000	0	52
MOMI Test	300	300	0	42
SOMI Test	200	200	0	100
SOMI-Real Test	200	0	200	16

The 20,000 MOMI training records are derived from 13,351 unique source–target image pairs. Multiple instruction-conditioned records may therefore share the same source–target pair.

Table 8: Record structure of the MOMI training set.

Structural Dimension	Count
Instruction-conditioned records	20,000
Unique source–target image pairs	13,351
Unique conditional image files	3,842
Unique target image files	3,844
Unique text instructions	18,856

Edit and Instruction Distributions

MOMI Train and MOMI Test cover four edit types, while SOMI Test and SOMI-Real are balanced between single-category addition and removal (Tables 9 and 10). Across MOMI Train and MOMI Test, instructions target a single object category in approximately 42% of records (41.5% in Train, 42.7% in Test) and two object categories in 58% of records (58.5% in Train, 57.3% in Test). MOMI additionally covers six approximately balanced instruction-complexity configurations, while SOMI and SOMI-Real contain only basic instructions (Tables 11 and 12).

Table 9: Edit-type distribution of the MOMI training set.

Edit Type	MOMI Train Count	Percentage
Multi-Object Same-Action	6,451	32.26%
Combined Remove+Add	5,247	26.24%
Single Addition	4,158	20.79%
Single Removal	4,144	20.72%

Table 10: Edit-type distribution across the evaluation benchmarks.

Edit Type	MOMI Test	SOMI Test	SOMI-Real
Multi-Object Same-Action	109 (36.33%)	0	0
Combined Remove+Add	63 (21.00%)	0	0
Single Addition	63 (21.00%)	100 (50.00%)	100 (50.00%)
Single Removal	65 (21.67%)	100 (50.00%)	100 (50.00%)

Table 11: Instruction-complexity distribution of the MOMI training set.

Instruction Type	MOMI Train Count	Percentage
Basic	3,272	16.36%
Color	3,329	16.64%
Spatial	3,377	16.89%
Color-Spatial	3,389	16.95%
Detailed-Spatial	3,362	16.81%
Color Detailed-Spatial	3,271	16.36%

Table 12: Instruction-complexity distribution across the evaluation benchmarks.

Instruction Type	MOMI Test	SOMI Test	SOMI Real
Basic	48 (16.00%)	200 (100.00%)	200 (100.00%)
Color	56 (18.67%)	0	0
Spatial	51 (17.00%)	0	0
Color-Spatial	48 (16.00%)	0	0
Detailed-Spatial	50 (16.67%)	0	0
Color Detailed-Spatial	47 (15.67%)	0	0

Color and Spatial Components

A color component denotes one requested color–object binding, while a spatial component denotes one requested object–location binding. SOMI and SOMI-Real contain only basic instructions and therefore do not contain explicitly requested color or spatial components.

Table 13: Requested color and spatial component distributions for MOMI Train and MOMI Test.

(a) Color components

Components	MOMI Train	MOMI Test
0	10,011 (50.06%)	149 (49.67%)
1–3	6,642 (33.21%)	79 (26.33%)
4–6	2,780 (13.90%)	54 (18.00%)
7–12	556 (2.78%)	16 (5.33%)
13+	11 (0.06%)	2 (0.67%)

(b) Spatial components

Components	MOMI Train	MOMI Test
0	6,601 (33.01%)	104 (34.67%)
1–3	2,213 (11.06%)	36 (12.00%)
4–6	3,859 (19.30%)	52 (17.33%)
7–12	5,657 (28.29%)	82 (27.33%)
13+	1,670 (8.35%)	26 (8.67%)

Requested Edit Quantity

The requested edit quantity, N_{req}, is the total number of object instances requested for addition or removal. Summary statistics and binned distributions across training and evaluation splits are provided in Tables 14, 15, and 16.

Table 14: Summary statistics of requested edit quantities.

Statistic	MOMI Train	MOMI Test	SOMI Test	SOMI Real
Mean	12.49	12.37	5.50	6.78
Median	12.0	12.0	5.5	7.0
Minimum	1	1	1	1
Maximum	30	28	10	15

Table 15: Binned distribution of requested edit quantities for the MOMI training set.

N_{req}	MOMI Train Count	Percentage
1–3	2,045	10.22%
4–6	2,102	10.51%
7–12	6,184	30.92%
13–18	5,756	28.78%
19–24	2,941	14.70%
25+	972	4.86%

Table 16: Binned distribution of requested edit quantities across evaluation benchmarks.

N_{req}	MOMI Test	SOMI Test	SOMI-Real
1–3	33 (11.00%)	60 (30.00%)	48 (24.00%)
4–6	27 (9.00%)	60 (30.00%)	49 (24.50%)
7–12	100 (33.33%)	80 (40.00%)	89 (44.50%)
13–18	80 (26.67%)	0	14 (7.00%)
19–24	45 (15.00%)	0	0
25+	15 (5.00%)	0	0

Object Edit Ratio Distribution and Shortcut Bias Mitigation

During initial model exploration, we observed a shortcut bias: when training datasets contain a high proportion of edits that remove a large fraction ($> 50\%$) or all ($i = N$) instances of a target category from a scene, text-guided image editing models frequently learn a trivial shortcut—blindly removing every instance of the requested object category regardless of the numerical count specified in the prompt (e.g., clearing all objects even when instructed to remove only two or three instances).

To mitigate this failure mode, we use two complementary mechanisms. First, during candidate generation, we exclude removal edits that remove all target instances ($i = N$), ensuring that at least one target instance remains after every removal operation. Because the remaining edit types are constructed from these verified removal edits through reversal or composition, this constraint also prevents full target-category edits in Single Addition, Remove+Add, and Multi-Object Same-Action examples. Second, from the larger pool of generated and verified edits, we apply a sampling strategy across all four edit types to obtain a broad and balanced distribution of edit ratios.

As shown in Table 17, our sampling strategy produces similar edit-ratio distributions across all four edit types, spanning the full range of partial edits without concentrating the dataset around large-fraction edits. No final training records fall in the 100% bin, both because full removals are excluded during candidate generation and because all other edit types are derived from these partial-removal edits.

Table 17: Decile distribution of object edit percentages across the four edit types in MOMI Train after dataset sampling.

Edit Ratio Bin	Single Addition ($N = 4{,}158$)	Single Removal ($N = 4{,}144$)	Remove+Add ($N = 5{,}247$)	Multi-Object ($N = 6{,}451$)
0–10%	403 (9.69%)	413 (9.97%)	423 (8.06%)	493 (7.64%)
10–20%	524 (12.60%)	557 (13.44%)	615 (11.72%)	692 (10.73%)
20–30%	403 (9.69%)	435 (10.50%)	613 (11.68%)	628 (9.73%)
30–40%	494 (11.88%)	496 (11.97%)	589 (11.23%)	792 (12.28%)
40–50%	584 (14.05%)	530 (12.79%)	705 (13.44%)	988 (15.32%)
50–60%	461 (11.09%)	436 (10.52%)	597 (11.38%)	718 (11.13%)
60–70%	460 (11.06%)	467 (11.27%)	573 (10.92%)	800 (12.40%)
70–80%	394 (9.48%)	370 (8.93%)	485 (9.24%)	692 (10.73%)
80–90%	322 (7.74%)	328 (7.92%)	498 (9.49%)	505 (7.83%)
90–$< 100\%$	113 (2.72%)	112 (2.70%)	149 (2.84%)	143 (2.22%)
100%	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
Total Records	**4,158 (100.0%)**	**4,144 (100.0%)**	**5,247 (100.0%)**	**6,451 (100.0%)**

Source Scene Density

Source scene density is the number of target-category object instances present in the source image before editing.

Table 18: Summary and distribution of source scene density.

(a) Summary statistics

Statistic	MOMI	SOMI	Real
Mean	25.08	9.06	9.63
Median	26.0	9.0	11.0
Minimum	2	0	1
Maximum	44	30	24

(b) Binned distribution

Source Count	MOMI	SOMI	Real
1–3	8 (2.67%)	17 (8.50%)	30 (15.00%)
4–6	10 (3.33%)	41 (20.50%)	25 (12.50%)
7–12	49 (16.33%)	104 (52.00%)	100 (50.00%)
13–18	68 (22.67%)	32 (16.00%)	41 (20.50%)
19–24	53 (17.67%)	6 (3.00%)	4 (2.00%)
25+	112 (37.33%)	0	0

OBJECT CATEGORY FREQUENCIES

MOMI Train contains 52 target categories and 31,698 target-category occurrences. MOMI Test contains 42 target categories and 472 target-category occurrences. Their union contains 62 categories, with 32 shared categories. SOMI Test contains 100 categories with exactly two samples per category, while SOMI-Real contains 16 categories distributed across 200 samples. Their union contains 109 categories, with seven shared categories.

Table 19: Target-category frequencies for MOMI Train and MOMI Test (part 1 of 2).

Category	Train	Test
apple	3,076	31
backpack	85	6
balloon	1,623	0
banana	751	0
battery	490	4
belt	138	3
bird	1,704	21
bowl	556	13
branch	50	0
button	1,176	17
calculator	317	3
camera	173	20
coconut	1,236	24
corkscrew	7	0
crow	166	0
cushion	193	12
donut	1,609	0
egg	1,928	31
eggplant	12	0
flashlight	36	0
goat	1,671	34
guitar	20	0
gummy	336	0
hat	4	0
headphone	0	3
hinge	198	4
keyboard	0	6
knife	160	11
ladle	0	2
laptop	105	2
microphone	0	8

Table 20: Target-category frequencies for MOMI Train and MOMI Test (part 2 of 2).

Category	Train	Test
microwave	0	11
mouse	31	6
mug	74	0
notebook	409	10
okra	427	1
olive	355	0
onion	1,745	17
orange	1,168	16
peach	871	0
pen	375	4
pillow	332	4
plant	1	0
plunger	0	2
pomegranate	324	5
potato	113	3
pumpkin	181	7
rabbit	6	0
radar	212	0
radio	0	3
refrigerator	0	11
shoe	99	12
soap	0	2
sock	50	12
spoon	2,925	38
stone	5	0
strawberry	417	1
teapot	39	0
tomato	1,935	15
trashcan	1,626	35
tv	0	2
wheel	158	0

Table 21: Target-category frequencies for SOMI Test and SOMI-Real Test (part 1 of 4).

Category	SOMI	Real
acorn	2	0
armband	2	0
ashtray	2	0
asparagus	2	0
bagel	2	0
ball	0	27
balloon	2	0
basketball	2	0
beaker	2	0
bird	0	2
blackberry	0	4
boat	2	0
boot	2	0
bottle	2	0
bracket	2	0
brownie	0	18
bun	0	2
button	2	0
cake	2	0
camera	2	0
candle	2	0
candy	2	0
cap	2	0
car	0	19
cat	2	0
caterpillar	2	0
cauliflower	2	0
chair	2	0

Table 22: Target-category frequencies for SOMI Test and SOMI-Real Test (part 2 of 4).

Category	SOMI	Real
cherry	2	14
chicken	0	18
chip	2	0
cleat	2	0
clock	2	0
clothespin	2	0
collet	2	0
cookie	2	8
corkscrew	2	0
cushion	2	0
desk	2	0
dice	2	0
disc	2	0
doll	2	0
donut	2	18
drumstick	2	0
duck	2	0
durian	2	0
egg	2	27
fish	2	0
flashlight	2	0
flower	0	10
folder	2	0
footstool	2	0
fuse	2	0
gem	2	0
goldfish	2	0
gummy	2	0

Table 23: Target-category frequencies for SOMI Test and SOMI-Real Test (part 3 of 4).

Category	SOMI	Real
hook	2	0
jar	2	0
kite	2	0
knife	2	0
lamppost	2	0
lantern	2	0
lightbulb	2	0
log	2	0
lug	2	0
lychee	2	0
meat	0	10
meeple	2	0
mitten	2	0
muffin	2	6
mug	2	0
mushroom	2	0
nail	2	0
okra	2	0
pearl	2	0
phone	2	0
pillow	2	0
pineapple	2	0
plunger	2	0
popcorn	2	0
potato	2	0
puck	2	0
radar	2	0
raspberry	2	0

Table 24: Target-category frequencies for SOMI Test and SOMI-Real Test (part 4 of 4).

Category	SOMI	Real
refrigerator	2	0
samosa	2	0
saucepan	2	0
seahorse	2	0
seed	2	0
shaker	2	0
shampoo	2	0
sheep	2	13
sock	2	0
sponge	2	0
stamp	2	0
stone	2	0
straw	2	0
strawberry	2	4
table	2	0
thimble	2	0
tire	2	0
token	2	0
tomato	2	0
trophy	2	0
tube	2	0
turban	2	0
violin	2	0
wallet	2	0
zucchini	2	0

CROSS-SPLIT CATEGORY OVERLAP

The low category overlap between MOMI and the SOMI benchmarks supports the use of SOMI as an out-of-distribution category evaluation, while SOMI-Real additionally introduces a real-image domain shift.

Table 25: Pairwise overlap between target-category vocabularies.

Split Pair	$\|A\|$	$\|B\|$	Shared	Jaccard
MOMI Train / MOMI Test	52	42	32	0.5161
MOMI Train / SOMI	52	100	19	0.1429
MOMI Train / SOMI-Real	52	16	4	0.0625
MOMI Test / SOMI	42	100	13	0.1008
MOMI Test / SOMI-Real	42	16	3	0.0545
SOMI / SOMI-Real	100	16	7	0.0642

Shared Categories MOMI Train and MOMI Test share: apple, backpack, battery, belt, bird, bowl, button, calculator, camera, coconut, cushion, egg, goat, hinge, knife, laptop, mouse, notebook, okra, onion, orange, pen, pillow, pomegranate, potato, pumpkin, shoe, sock, spoon, strawberry, tomato, and trashcan. MOMI Train and SOMI share: balloon, button, camera, corkscrew, cushion, donut, egg, flashlight, gummy, knife, mug, okra, pillow, potato, radar, sock, stone, strawberry, and tomato. MOMI Train and SOMI-Real share bird, donut, egg, and strawberry. MOMI Test and SOMI share button, camera, cushion, egg, knife, okra, pillow, plunger, potato, refrigerator, sock, strawberry, and tomato. MOMI Test and SOMI-Real share bird, egg, and strawberry. SOMI and SOMI-Real share cherry, cookie, donut, egg, muffin, sheep, and strawberry.

All dataset-level statistics in this section are generated programmatically from the structured ground-truth metadata. The verification scripts, dataset hashes, and assertion logs are included with the supplementary dataset release.

.3 QUALITATIVE DATASET EXAMPLES

This section provides representative qualitative dataset examples across all splits of our benchmark suite: MOMI Train, MOMI Test, SOMI Test, and SOMI-Real Test. Figures 6–18 illustrate diverse edit operations (*Single Addition*, *Single Removal*, *Multi-Object Same-Action*, and *Combined Remove+Add*) alongside varied instruction granularities ranging from basic quantity instructions to fine-grained color and spatial constraints. Each figure panel displays the side-by-side source and target images accompanied by a prompt caption card formatted as `[Instruction Type | Edit Type]` followed by the editing prompt.

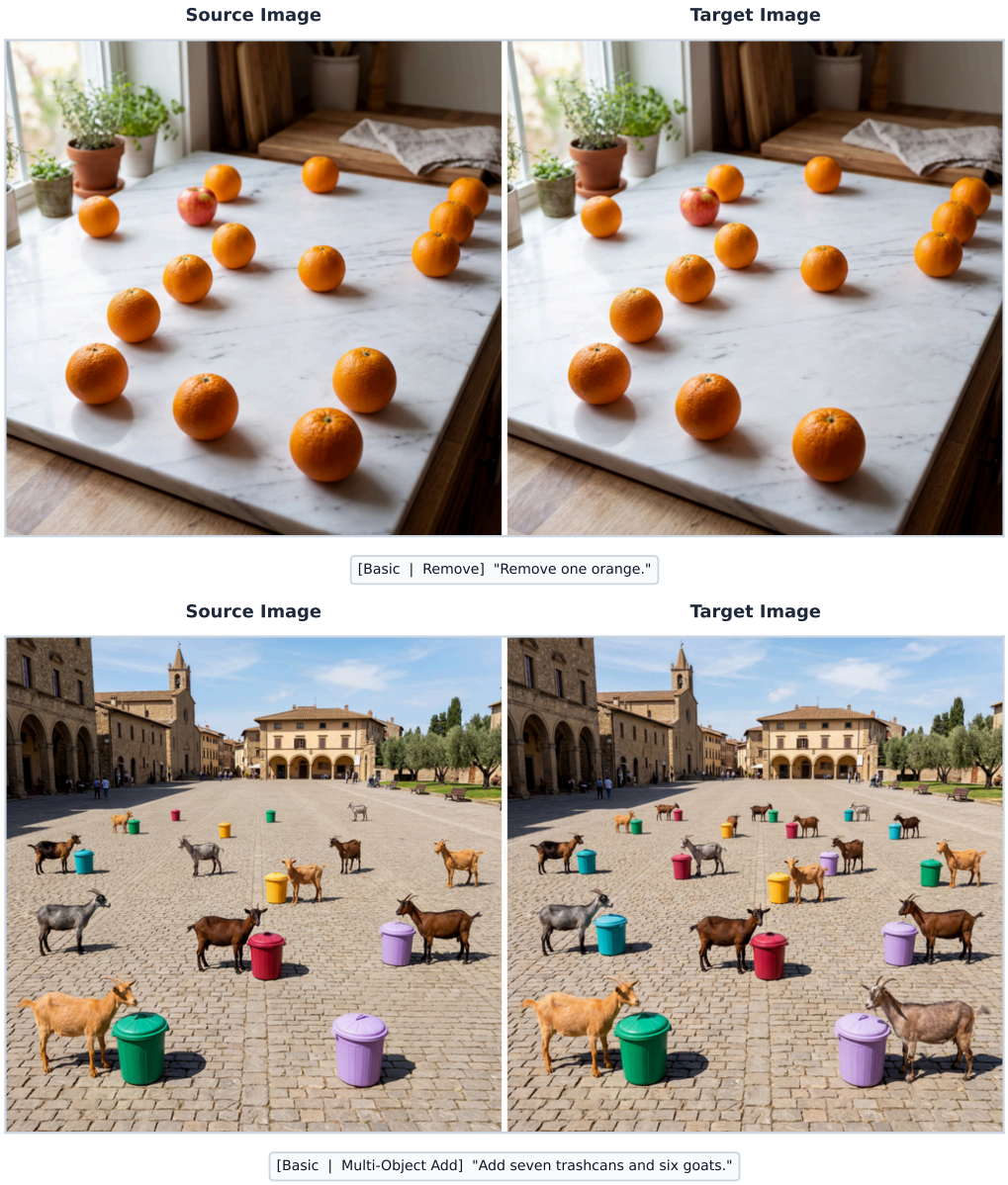
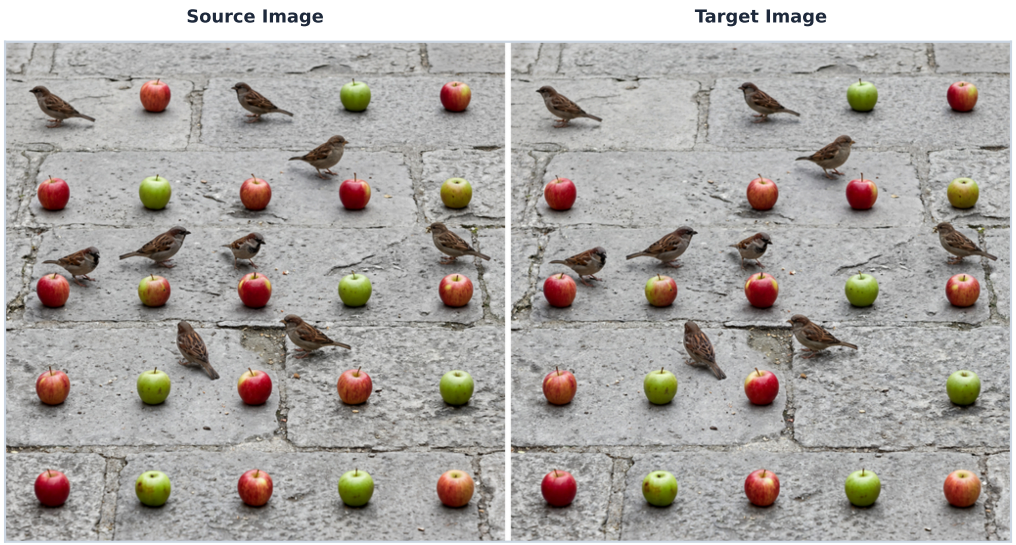
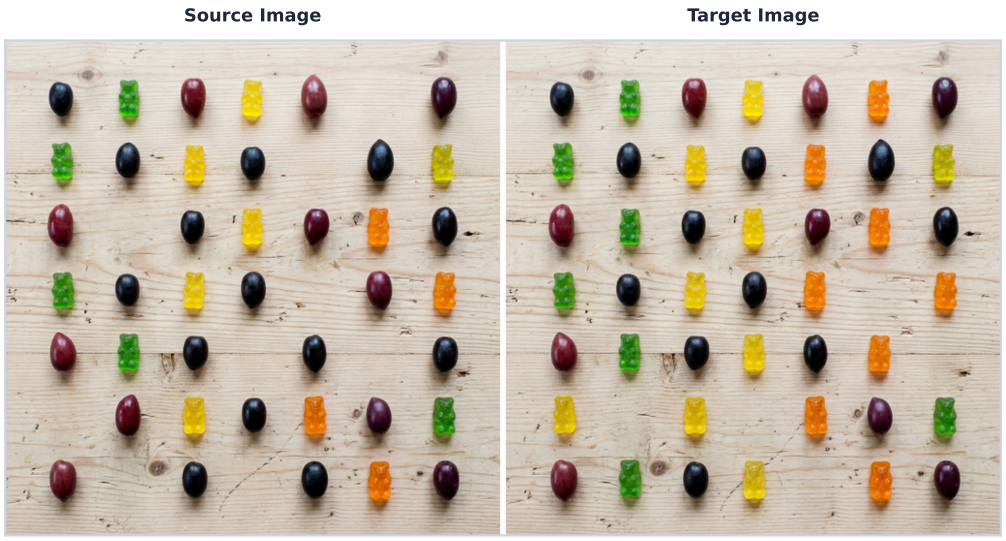


Figure 6: Qualitative dataset examples from the **MOMI Train** split (Part 1 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.



[Color-Spatial | Multi-Object Remove] "Remove red apple in top left, green apple in middle left, and red apple in lower right."



[Color-Detailed-Spatial | Remove + Add] "Add four orange gummies located near top right corner, in upper right area, in exact middle right, and in lower right section, three yellow gummies along lower left edge, right in lower center, and at bottom center edge, and two green gummies on upper middle-left side and in bottom left area and remove purple olives located in lower left section and middle right area, and black olives in exact lower center, on rightmost edge below middle, and in bottom row slightly right of center."

Figure 7: Qualitative dataset examples from the **MOMI Train** split (Part 2 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.

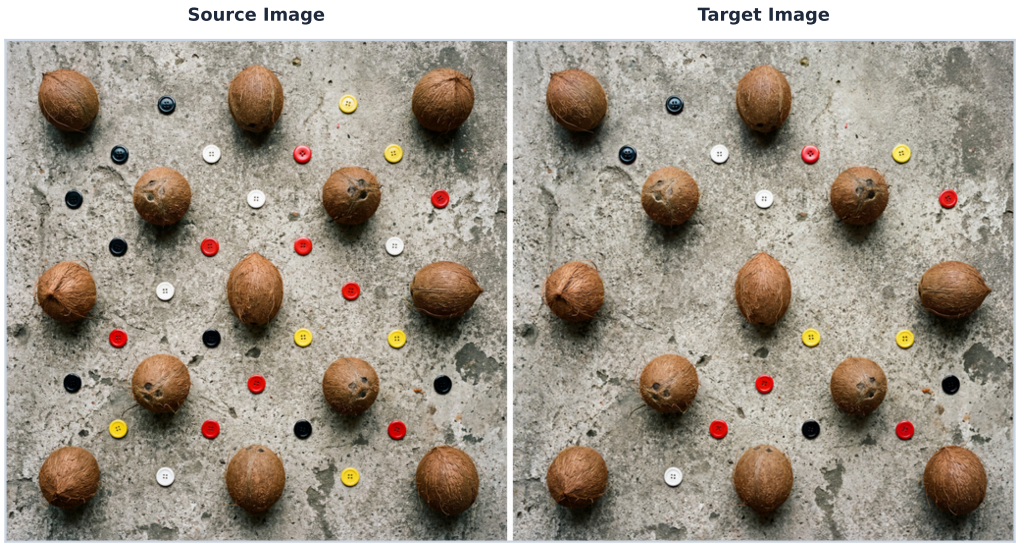
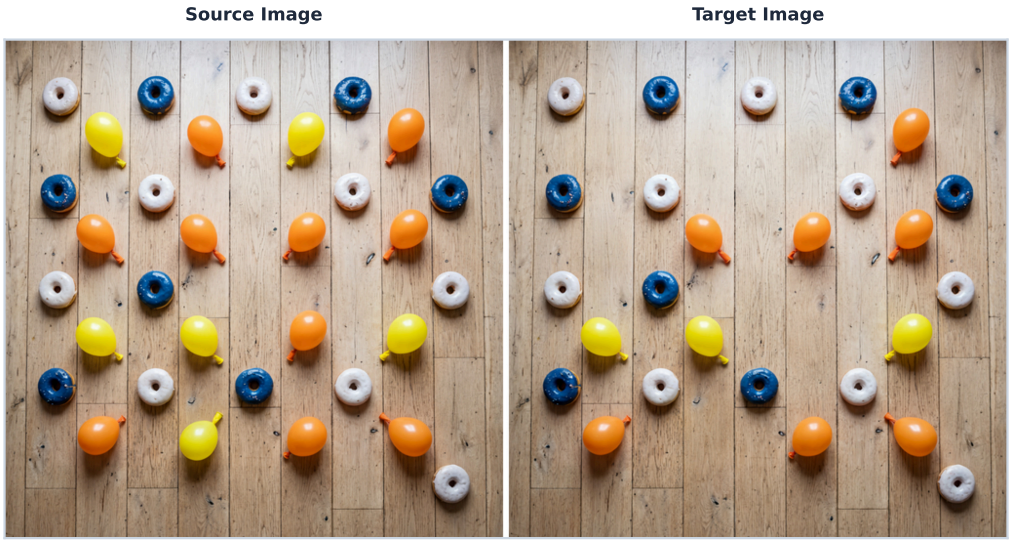
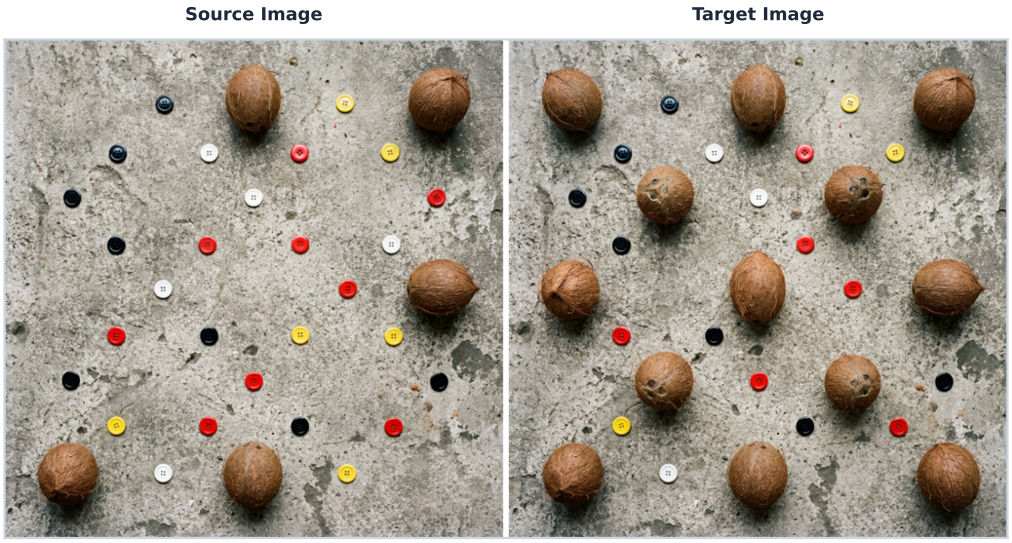


Figure 8: Qualitative dataset examples from the **MOMI Train** split (Part 3 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.



[Color-Detailed-Spatial | Multi-Object Remove] "Remove yellow balloon in top left corner, orange balloon near middle left edge, orange balloon in right area slightly below center, yellow balloon in bottom area slightly left of center, and orange and yellow balloons spread across upper middle area."



[Spatial | Remove + Add] "Add eight coconuts in top left, middle left, center, bottom right, and four located diagonally around center and remove nine buttons in lower left, middle left, upper middle left, lower middle left, upper right, upper middle right, bottom right, and two in middle right."

Figure 9: Qualitative dataset examples from the **MOMI Train** split (Part 4 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.



Figure 10: Qualitative dataset examples from the **MOMI Test** benchmark split (Part 1 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.



Figure 11: Qualitative dataset examples from the **MOMI Test** benchmark split (Part 2 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.

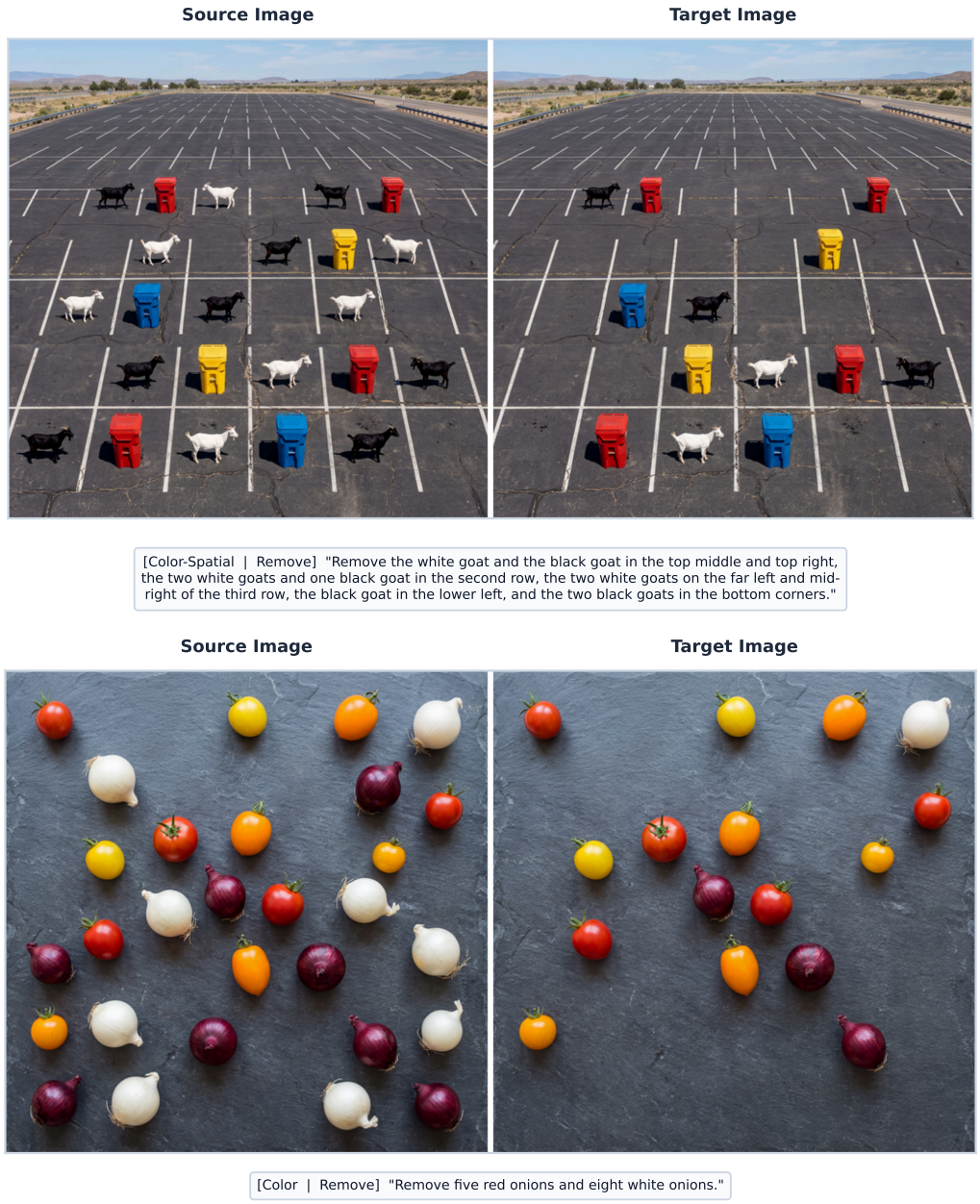


Figure 12: Qualitative dataset examples from the **MOMI Test** benchmark split (Part 3 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.



Figure 13: Qualitative dataset examples from the **MOMI Test** benchmark split (Part 4 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.

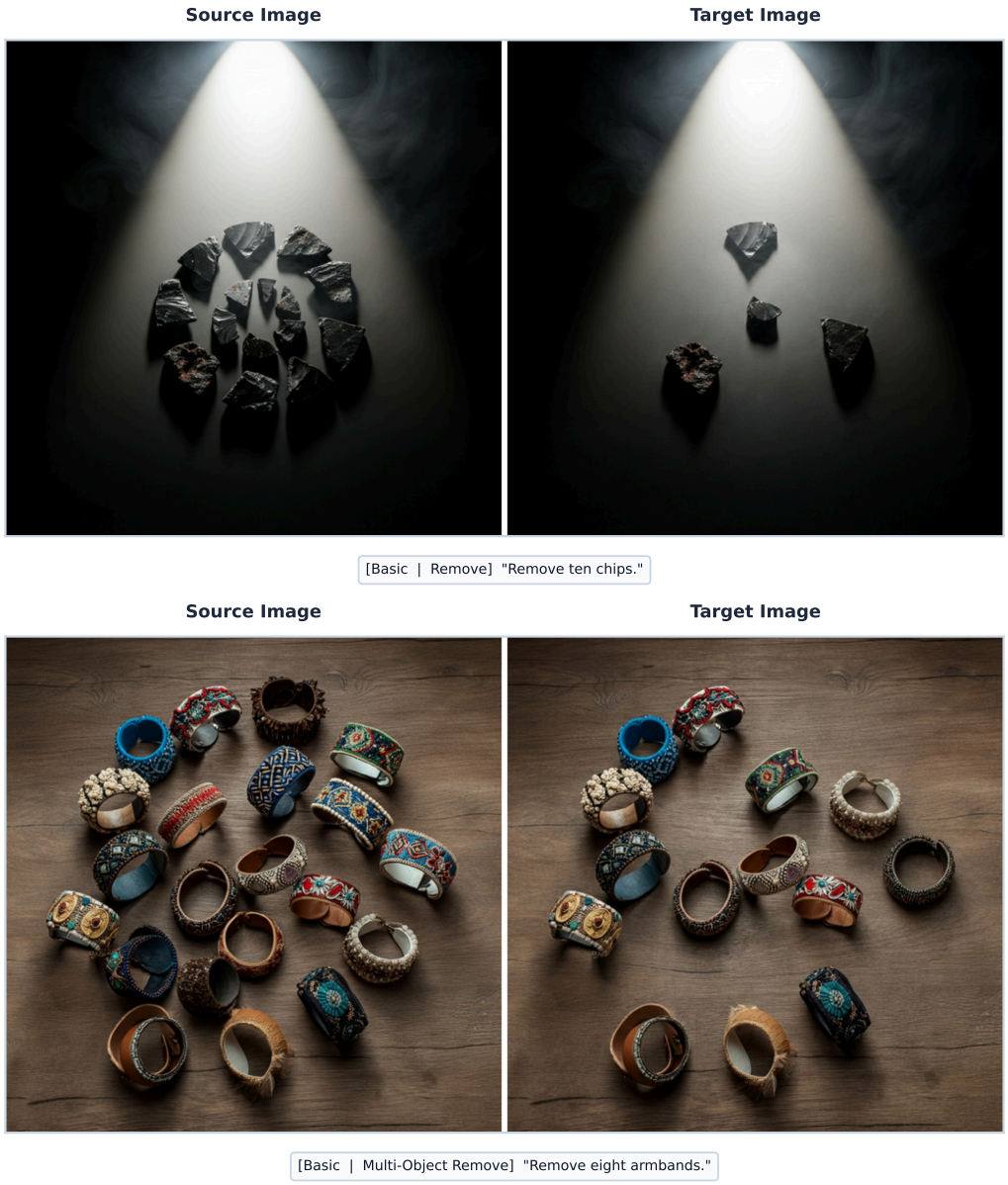


Figure 14: Qualitative dataset examples from the **SOMI Test** synthetic benchmark split (Part 1 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.

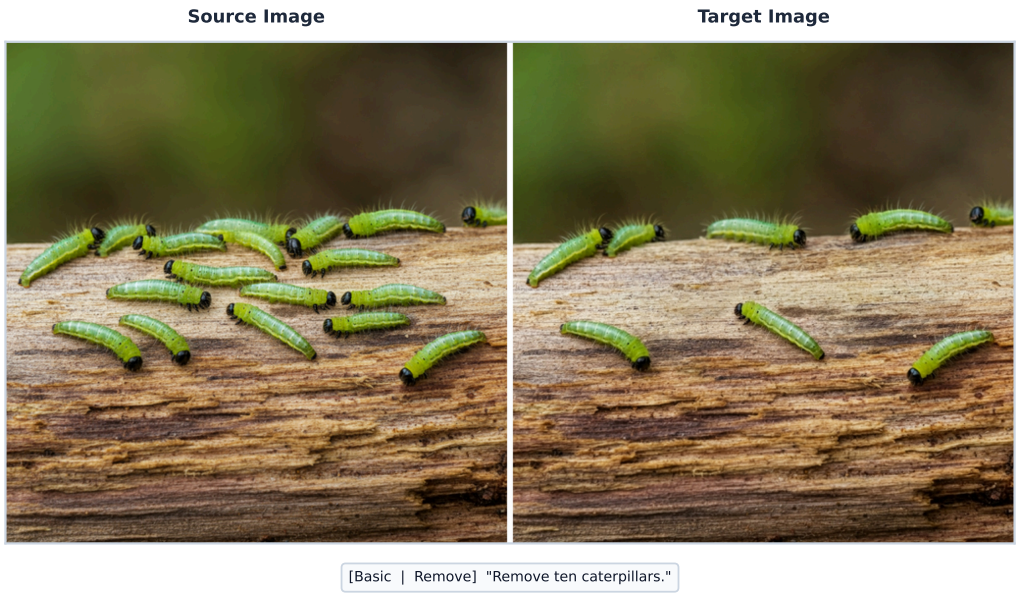


Figure 15: Qualitative dataset examples from the **SOMI Test** synthetic benchmark split (Part 3 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.

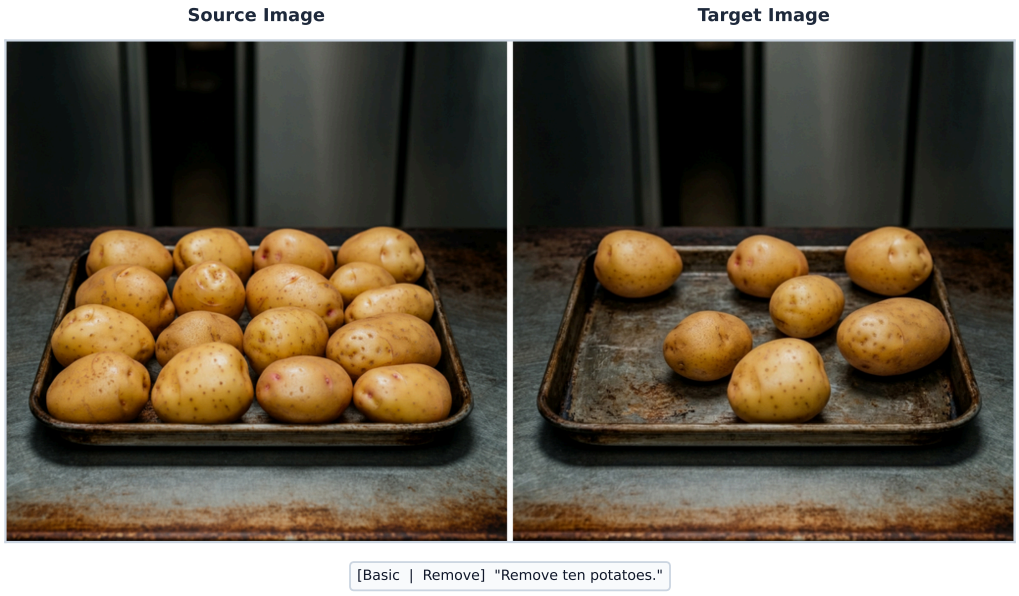


Figure 16: Qualitative dataset examples from the **SOMI Test** synthetic benchmark split (Part 4 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.



Figure 17: Qualitative dataset examples from the **SOMI-Real Test** photograph benchmark split (Part 1 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.

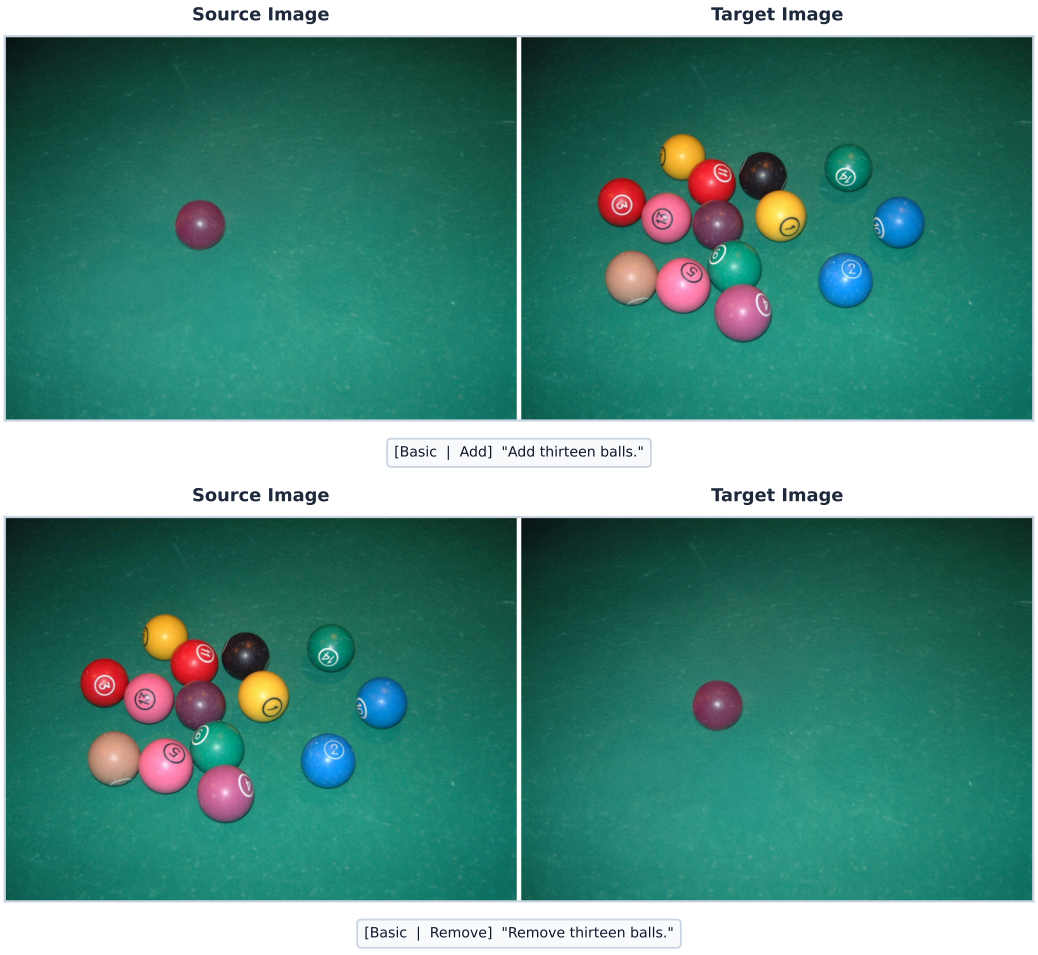


Figure 18: Qualitative dataset examples from the **SOMI-Real Test** photograph benchmark split (Part 2 of 4). Each example presents side-by-side source and target images with a prompt caption card in the `[Instruction Type | Edit Type]` format.

.4 Human Validation of the Automated Evaluation Pipeline

We conduct two complementary human studies to validate the automated evaluation pipeline used to evaluate model-generated edits. The first evaluates the pipeline on the single-category SOMI benchmark using three independent annotators. The second provides targeted expert validation on the more complex MOMI benchmark, including multi-category, compound, and attribute-conditioned edits. Since the evaluation pipeline shares the same SAM 3-based instance counting mechanism used during dataset verification, these studies also provide supporting evidence for this core verification component.

All annotators were provided with detailed annotation instructions and were required to pass a qualification test before participating.

SOMI Annotation Protocol. We evaluate all 200 SOMI samples for BAGEL, Qwen Image Edit, and Flux Kontext, resulting in 600 model outputs. For each output, annotators were shown the source image, the editing instruction, and the model-generated output. They were asked to determine whether the output exactly satisfied the requested edit type, target object, and object count. An output was labeled as successful only when all three components were correct. Since SOMI contains Basic single-category multi-instance instructions, color and spatial attributes were not considered in this evaluation.

Each output was independently evaluated by three human annotators, and the majority vote was used as the human reference label. The binary decision produced by the automated evaluation pipeline was then compared with the human majority label at the sample level.

MOMI Expert Annotation Protocol. To validate the evaluator on more complex multi-instance edits, we select 100 MOMI-Test examples using stratified sampling to approximately preserve the benchmark distribution across the six instruction types and four edit types. We evaluate PGAMIE and Nanobanana 2 on the same 100 examples, yielding 200 model outputs. A human expert with research experience in image editing was shown the source image, editing instruction, and model-generated output.

For each output, the expert judged whether the requested edit action or actions, object category or categories, and object count or counts were correct. For instructions containing color attributes, the expert additionally judged whether the correct colored object instances were edited. Across the two models, 102 outputs contained color attributes.

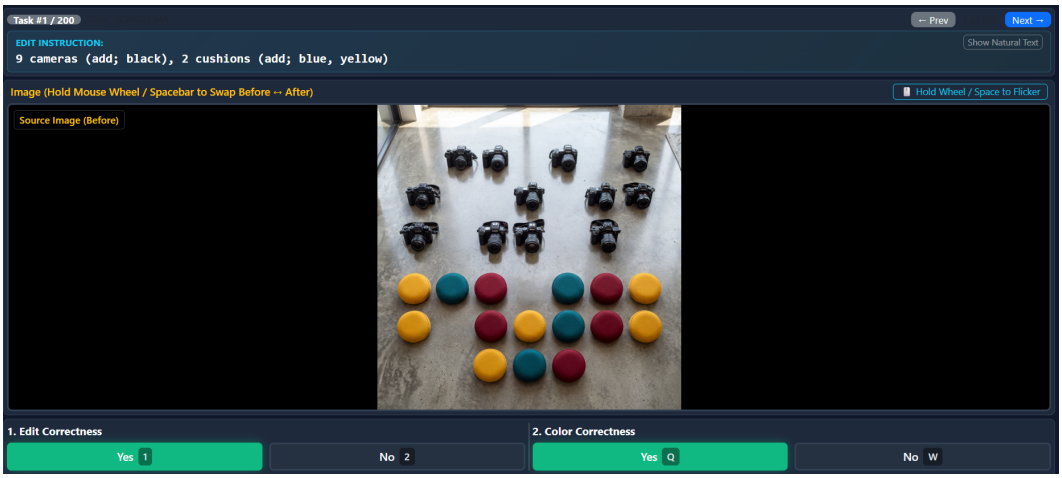


Figure 19: Human annotation interface. The annotator is shown the source image, editing instruction and evaluates edit correctness and, when applicable, color correctness. When pressing mouse wheel the source image swaps to the edited image.

Evaluation Metrics. For Edit Accuracy, we report raw agreement, Cohen's κ, precision, recall, and F_1 score, treating the human judgment as the reference label. A positive prediction indicates that the automated pipeline classified an output as an exact edit type–object–count match. For SOMI,

Model	κ	Agre	Pre	Rec	F_1	Human/Auto Acc.
Qwen Image Edit	0.792	96.5%	75.0%	88.2%	81.1%	8.5%/10.0%
Flux Kontext	0.693	96.0%	83.3%	62.5%	71.4%	8.0%/6.0%
BAGEL	0.615	96.0%	63.6%	63.6%	63.6%	5.5%/5.5%

Table 26: Agreement between human majority judgments and the automated evaluation pipeline on the SOMI benchmark. Agre, Pre, and Rec denote raw agreement, precision, and recall, respectively. Human/Auto Acc. denotes model-level Edit Accuracy under human and automated evaluation, respectively.

Model	TP	FP	FN	TN
Qwen Image Edit	15	5	2	178
Flux Kontext	10	2	6	182
BAGEL	7	4	4	185

Table 27: Sample-level confusion matrices of the automated evaluation pipeline relative to human majority judgments on SOMI. A positive label denotes an exact edit type–object–count match.

we additionally compare model-level Edit Accuracy obtained using the automated pipeline with the corresponding accuracy obtained from human majority judgments. For MOMI, we also report agreement between the automated color evaluation and the expert color judgment.

Results. On SOMI, the automated evaluation pipeline achieves 96.0–96.5% raw agreement with human majority judgments across the three evaluated models. Because successful edits constitute only 5.5–8.5% of the human labels, raw agreement may be influenced by class imbalance. Cohen's κ ranges from 0.615 to 0.792, indicating substantial chance-corrected agreement. Model-level Edit Accuracy differs from human evaluation by at most 2.0 percentage points.

On MOMI, the automated Edit Accuracy decision agrees with the expert on 192 of 200 outputs (96.0%), with Cohen's $\kappa = 0.839$, precision of 80.6%, recall of 92.6%, and F_1 of 86.2%. The expert labels 27 of 200 outputs as correct, compared with 31 of 200 under the automated evaluator. For the 102 outputs containing color attributes, the automated color evaluation agrees with the expert on 89 outputs (87.3%).

Together, the SOMI and MOMI studies show that the automated evaluator closely reproduces human judgments in both the simpler single-category setting and the more complex MOMI setting. The MOMI study specifically validates the evaluator on multi-category, compound, and attribute-conditioned edits. Since the evaluation and dataset-verification pipelines share the same SAM 3-based instance counting mechanism, these results also provide supporting evidence for the reliability of this core component during scalable dataset generation and filtering. Nevertheless, automated decisions are not perfectly interchangeable with human judgments at the individual-sample level.

Validation of the Grounding Backend: SAM 3 vs. YOLO-World

To evaluate the choice of SAM 3 as the grounding and count-change backend, we compare it against YOLO-World, an open-vocabulary object detector, using the same 600 human-annotated SOMI outputs used in our human-validation study (200 samples $\times$ 3 editing models: BAGEL, Qwen Image Edit, and Flux Kontext). SOMI contains single-category Add and Remove instructions; thus, this experiment evaluates whether each backend correctly recovers the performed action and quantity for the supplied target category.

For each source image I_{src} and edited image I_{edit}, both backends estimate the number of instances of the target category x. The count difference

$$\Delta(x) = C_{\text{edit}}(x) - C_{\text{src}}(x)$$

determines the inferred action and quantity: $\Delta > 0$ indicates addition, $\Delta < 0$ indicates removal, and $|\Delta|$ gives the number of edited instances. We then determine whether the inferred action and quantity exactly match the requested edit and compare this binary decision with the human majority judgment. Because each of the 200 SOMI samples is evaluated by three editing models, we compute

Agreement	κ	Precision	Recall	F_1	Human/Auto Acc.
96.0%	0.839	80.6%	92.6%	86.2%	13.5%/15.5%

Table 28: Targeted expert validation of Edit Accuracy on 200 MOMI outputs from PGAMIE and Nanobanana 2. The confusion matrix contains 25 true positives, 6 false positives, 2 false negatives, and 167 true negatives.

Table 29: **Grounding-backend validation on 600 human-annotated SOMI outputs.** Agreement, positive-class F_1, and Cohen's κ compare the edit-correctness decisions induced by each grounding backend with human majority judgments. Brackets report 95% confidence intervals from 10,000 clustered bootstrap resamples over the 200 SOMI samples.

Grounding Backend	**Agreement (%)** ↑	F_1 **(%)** ↑	**Cohen's** κ ↑
SAM 3	**96.2** [94.2, 97.8]	**73.6** [58.8, 84.9]	**0.715** [0.562, 0.836]
YOLO-World	91.0 [87.8, 93.8]	15.6 [3.1, 29.8]	0.116 [-0.015, 0.263]
Paired Difference	*+5.2 pp* [2.2, 8.3]	*+57.9 pp* [40.3, 73.8]	–

95% confidence intervals using 10,000 clustered bootstrap resamples over the 200 underlying SOMI sample IDs.

As shown in Table 29, SAM 3 produces edit-correctness decisions that align substantially better with human judgments than YOLO-World. Although raw agreement is relatively high for both backends because only 44 of the 600 human judgments are positive, SAM 3 achieves an F_1 score of 73.6% compared with 15.6% for YOLO-World, with a paired improvement of 57.9 percentage points (95% CI: $[40.3, 73.8]$). Cohen's κ similarly increases from 0.116 to 0.715. The paired confidence intervals for both F_1 and agreement exclude zero.

Among the 44 human-positive outputs, YOLO-World misclassifies 39 as incorrect edits. In 33 of these failures, it predicts no count change ($\Delta = 0$); five infer a change in the opposite direction, and one overestimates the edit magnitude. These results support the use of SAM 3 as the grounding and count-change backend in our automated evaluation pipeline.

.5 Complementary Roles of SAM 3 and Gemini VQA

Case Studies: Complementary Counting Behavior. Figures 20 and 21 illustrate complementary behaviors of SAM 3 and Gemini VQA. SAM 3 provides precise instance-level localization, but can over-count semantic instances because of part–whole fragmentation, overlapping objects, or small and peripheral detections. Gemini VQA instead focuses more strongly on complete semantic objects, providing a complementary count without pixel-level grounding.

For example, for the query "truck," SAM 3 returns 17 grounded proposals by separating parts such as cabins and cargo containers and detecting small peripheral objects, whereas Gemini identifies 8 complete trucks. Similar behavior occurs across other object categories. For "clock," SAM 3 produces 7 proposals for 6 clocks by generating multiple detections on the top clock; for "plant," it produces 22 proposals for 10 potted plants by segmenting foliage clusters and branches separately. In cluttered scenes, SAM 3 also produces additional proposals for overlapping or partially visible objects: it returns 24 proposals for 15 mangoes and 17 proposals for 14 prominent sunflowers. These examples motivate combining SAM 3 localization with Gemini-based semantic count verification to reduce errors caused by fragmentation, overlap, and peripheral detections.

.6 Extended Performance Analyses and Visualizations

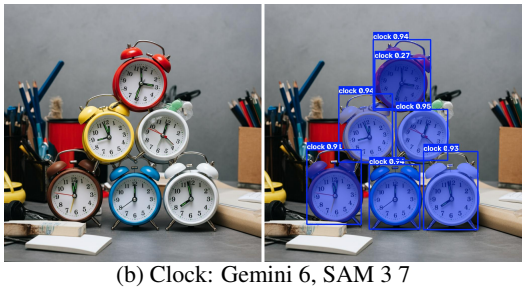
This appendix provides additional quantitative analyses supporting the findings in the main text. We report overall metrics, failure-mode analyses, results grouped by edit and instruction type, and fine-grained complexity analyses.



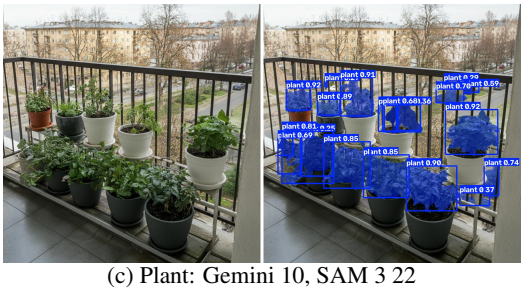
Figure 20: **Complementary behavior of SAM 3 and Gemini VQA. Left:** Original fleet image. **Right:** SAM 3 grounding for the query "truck," yielding 17 proposals because of part-level and small peripheral detections. Gemini VQA instead identifies 8 complete truck instances but provides no pixel-level grounding.



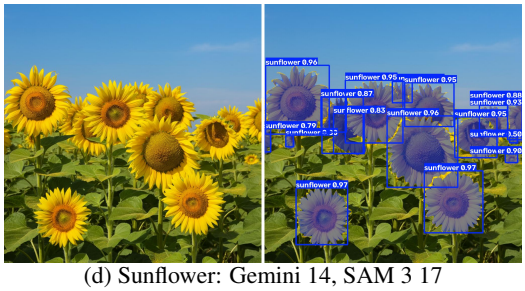
(a) Mango: Gemini 15, SAM 3 24



(b) Clock: Gemini 6, SAM 3 7



(c) Plant: Gemini 10, SAM 3 22



(d) Sunflower: Gemini 14, SAM 3 17

Figure 21: **Additional examples of complementary SAM 3 and Gemini VQA behavior.** SAM 3 can produce additional grounded proposals because of overlapping objects, part–whole fragmentation, and small or peripheral detections, while Gemini VQA focuses more strongly on complete semantic instances.

Overall Performance Metrics

Figures 22 and 23 display the Exact Edit Accuracy Match Rate and Exact Full-Instruction Match Rate with Soft Equivalence, respectively. Figures 24 and 25 show the Exact Color and Spatial Soft Match Rates, respectively.

Full-Instruction Exact Matching. When evaluation requires all instruction attributes to be satisfied jointly—including edit action, target object, exact count, color, and spatial location—performance remains low across all models, ranging from 0.3% to 7.3% overall on MOMI (Fig-

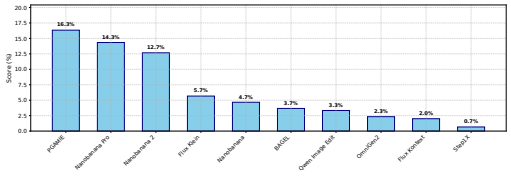


Figure 22: Edit Accuracy (Including Color and Spatial attributes)

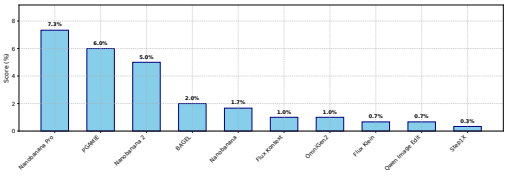


Figure 23: Exact Full-Instruction Match Rate (Soft Equivalence).

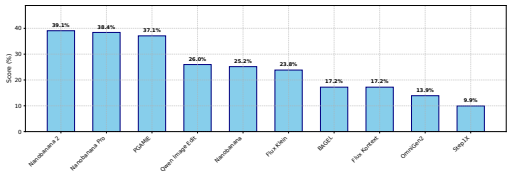


Figure 24: Exact Color Soft Match Rate.

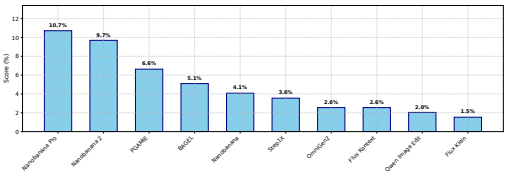


Figure 25: Exact Spatial Soft Match Rate.

ure 23). Nanobanana Pro achieves the highest match rate at 7.3%, followed by PGAMIE at 6.0% and Nanobanana 2 at 5.0%, while Flux Kontext and OmniGen2 reach 1.0% and Step1X 0.3%. By operation type, additions perform best (20.6% for Nanobanana Pro and 14.3% for PGAMIE and Nanobanana 2), whereas *Remove+Add* and multi-category instructions remain near zero (0.0%–2.8%). These results show that jointly resolving instance count, spatial grounding, and color binding remains challenging for current image editors.

Error and Artifact Analyses

To evaluate collateral damage and common failure modes, Figure 26 reports the No-Edit Rate. Figures 27 and 28 show the Extra Unrequested Edit Component Rate (capturing unrequested background modifications and spurious operations) and Wrong-Object-Only Error Rate, respectively. Figure 29 shows the Opposite-Action Error Rate (Add $\leftrightarrow$ Remove), while Figure 30 reports the Incomplete Add and Remove Edit Rate. Lower values indicate fewer editing failures for all reported metrics.

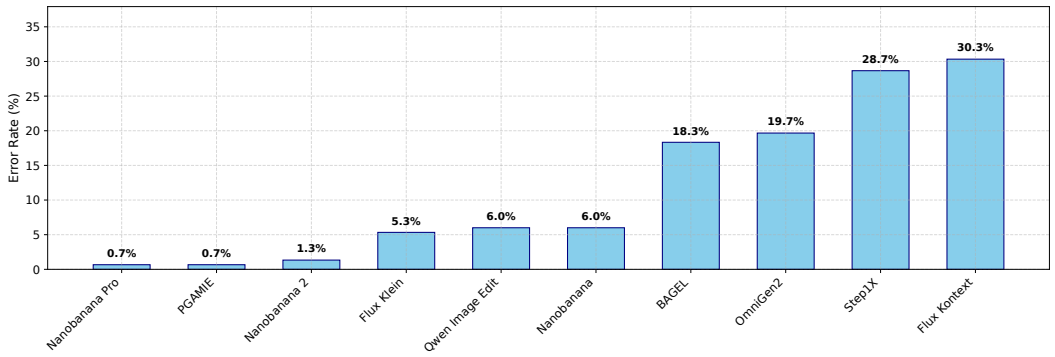


Figure 26: No Edit Occurred Rate across different models on the MOMI benchmark. Lower is better. The metric measures the percentage of valid evaluated samples in which no object addition or removal occurred. PGAMIE and Nanobanana achieve the lowest error rates, whereas Step1X and Flux Kontext produce no edit in roughly 30% of the samples.

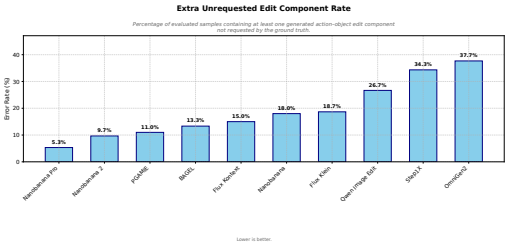


Figure 27: Extra Unrequested Edit Component Rate.

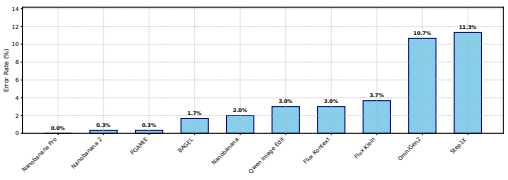


Figure 28: Wrong-Object-Only Error Rate.

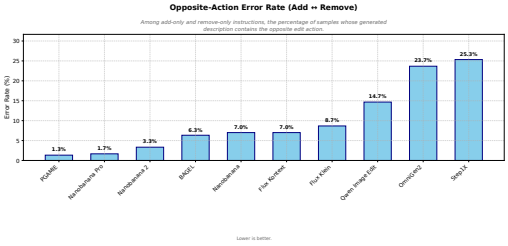


Figure 29: Opposite-Action Error Rate (Add ↔ Remove).

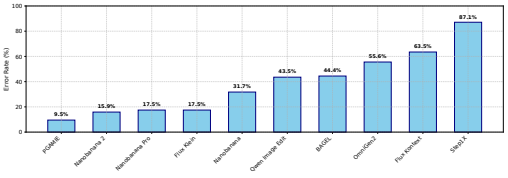


Figure 30: Incomplete Add and Remove Edit Rate.

METRICS GROUPED BY EDIT AND INSTRUCTION TYPE

Figures 31 and 32 display the Edit Accuracy Rate broken down by requested edit action (Addition vs. Removal) for each evaluated model on the SOMI (synthetic) and SOMI-Real benchmarks, respectively.

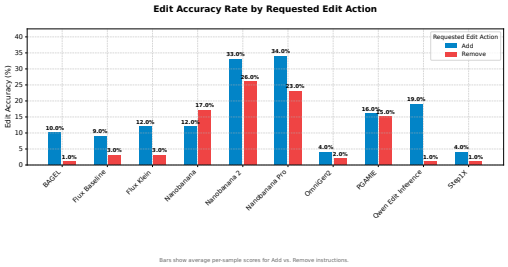


Figure 31: Edit Accuracy Rate by Requested Edit Action (Add vs. Remove) on the SOMI benchmark.

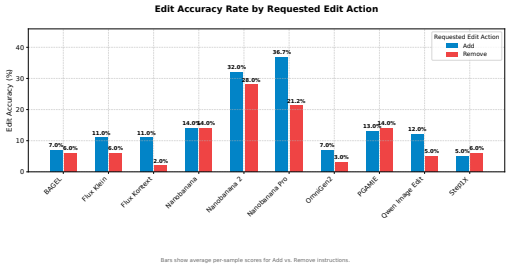


Figure 32: Edit Accuracy Rate by Requested Edit Action (Add vs. Remove) on the SOMI-Real benchmark.

Requested and Observed Edit-Type Heatmaps

Figures 33–44 compare the requested edit type with the observed edit classification. Rows denote the requested edit type, while columns denote the edit operation performed by the model. The pooled heatmaps summarize behavior across all evaluated models, and the remaining heatmaps show the corresponding distributions for each model separately.

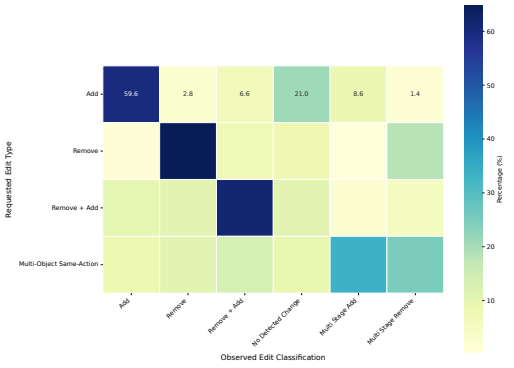


Figure 33: Requested versus observed edit types pooled across all models and normalized within each requested edit type.

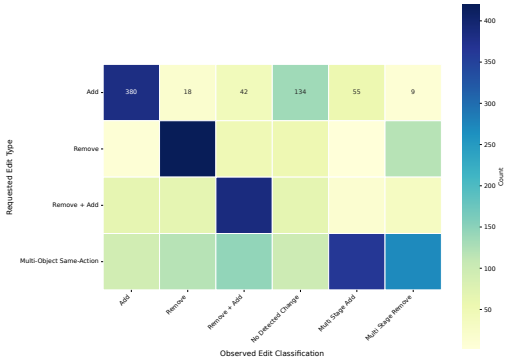


Figure 34: Raw requested-versus-observed edit-type counts pooled across all models.

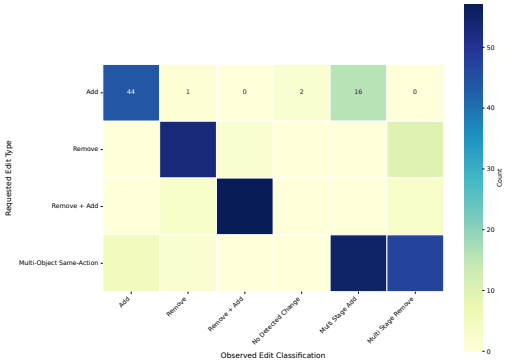


Figure 35: Requested versus observed edit types for PGAMIE.

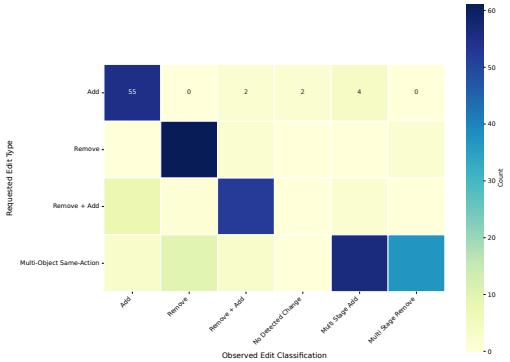


Figure 36: Requested versus observed edit types for Nanobanana Pro.

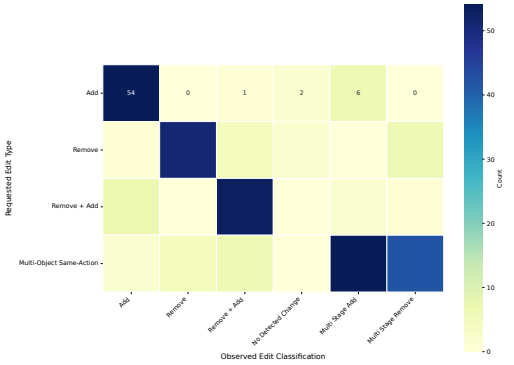


Figure 37: Requested versus observed edit types for Nanobanana 2.

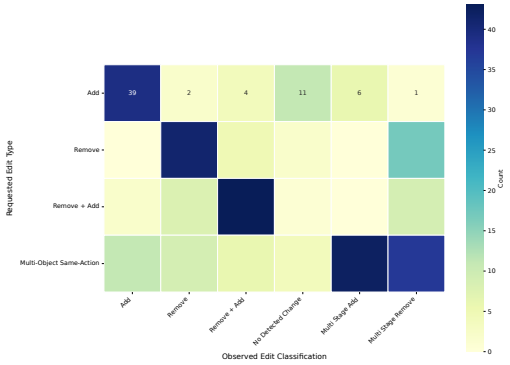


Figure 38: Requested versus observed edit types for Nanobanana.

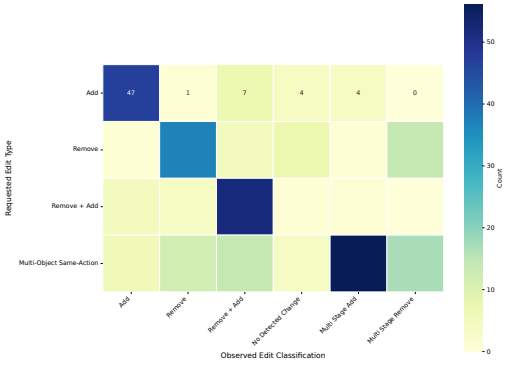


Figure 39: Requested versus observed edit types for Flux Klein.

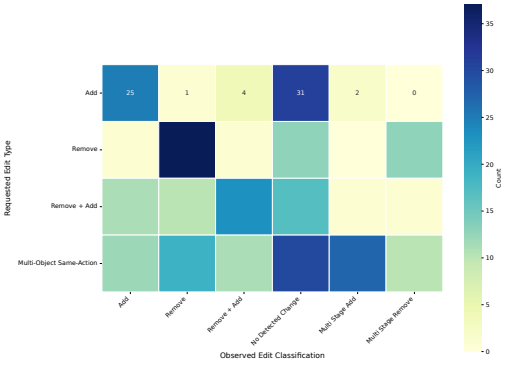


Figure 40: Requested versus observed edit types for Flux Kontext.

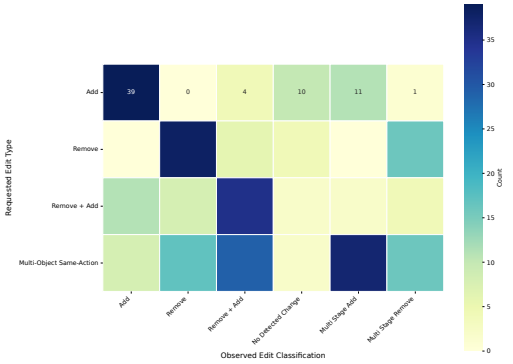


Figure 41: Requested versus observed edit types for Qwen Image Edit.

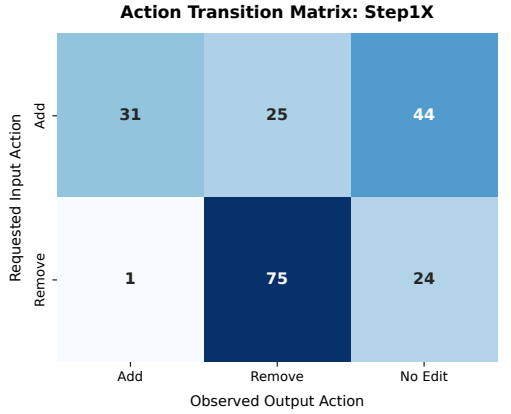


Figure 42: Requested versus observed edit types for Step1X.

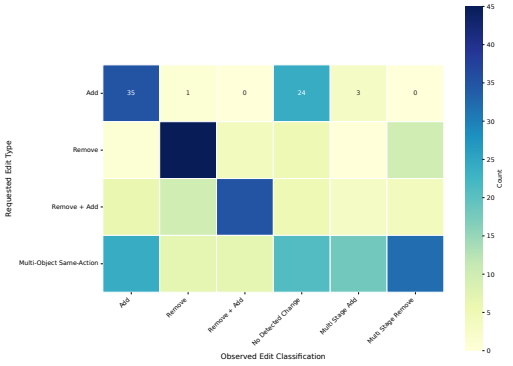


Figure 43: Requested versus observed edit types for BAGEL.

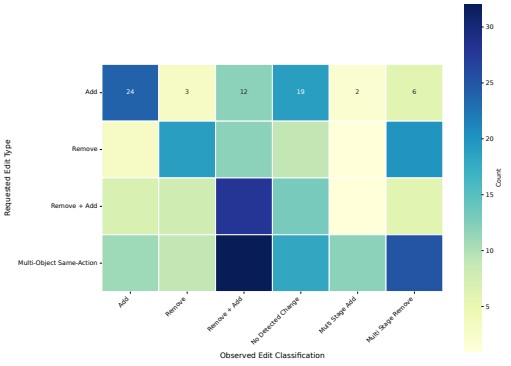


Figure 44: Requested versus observed edit types for OmniGen2.

Fine-Grained Complexity and Object Density Analyses

Figure 51 shows Edit Accuracy across color-complexity levels. Figure 50 show Edit accuracy across spatial-complexity levels. Figures 46–49 report Edit Accuracy as a function of scene object count, spatial occupancy, and requested edit count across different binning splits.

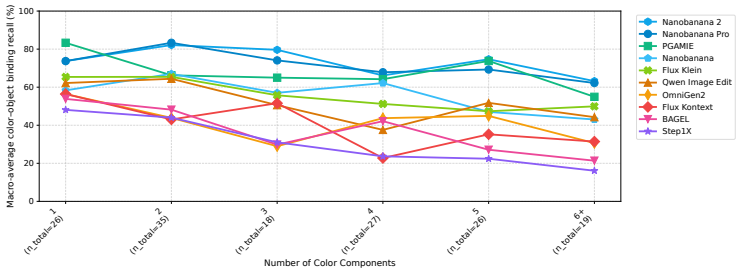


Figure 45: Color–Object Binding Recall by Color Complexity Level.

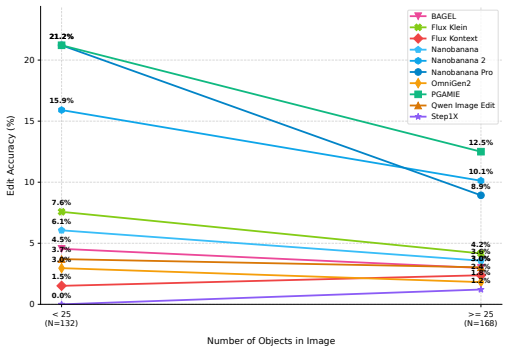


Figure 46: Edit Accuracy by Scene Object Count (Binary Split).

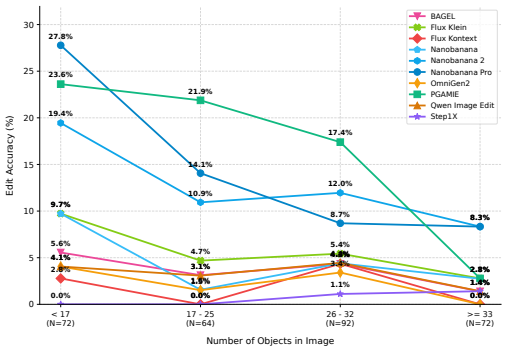


Figure 47: Edit Accuracy by Scene Object Count (Quartile Split).

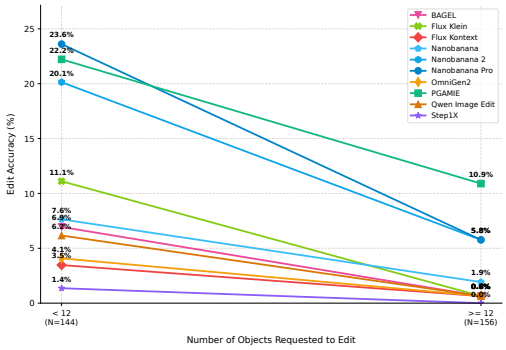


Figure 48: Edit Accuracy by Requested Edit Count (Binary Split).

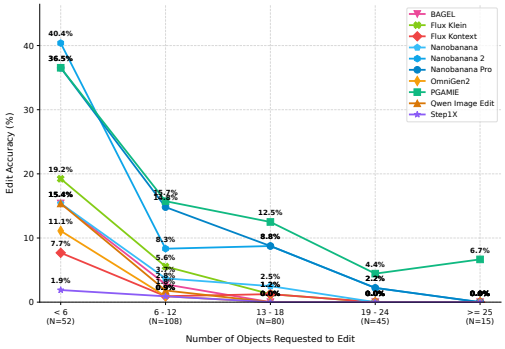


Figure 49: Edit Accuracy by Requested Edit Count (5-Bin Step-Wise Split).

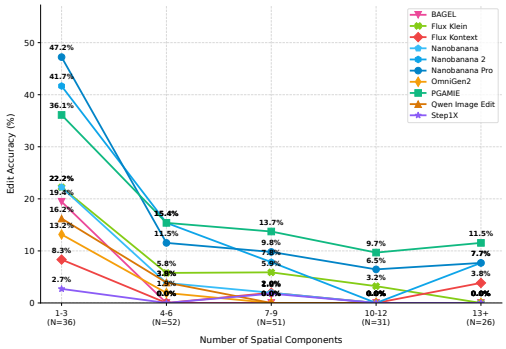


Figure 50: Edit Accuracy by Number of Spatial Components.

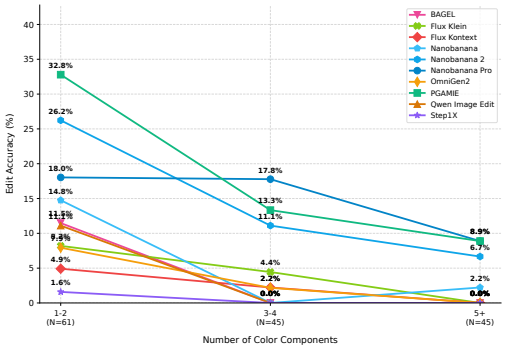


Figure 51: Edit Accuracy by Number of Color Components.

.7 Detailed Evaluation Methodology and Prompt Specifications

This section provides an end-to-end description of the two-step evaluation framework implemented in `step_1_generate_predictions.py` and `step_2_evaluate_predictions.py`, detailing the grounding pipeline, component extraction rules, formal metric definitions, and full prompt templates.

Pipeline Architecture and Step 1 Grounding

The evaluation pipeline operates in two stages:

1. **Step 1: Grounding and Edit Prediction Extraction (`step_1_generate_predictions.py`).** Given a source image I_A, an edited output image I_B, and the corresponding benchmark metadata, we first extract the object categories referenced by the ground-truth instruction and augment them with categories encoded in the source and target filenames. SAM 3 is then used to segment these categories in both images, producing instance masks, bounding boxes, and per-category object counts.

   Differences between the source and edited-image counts determine the observed edit type and the number of edited instances. To localize instance-level changes, source and target bounding boxes belonging to the same category are matched using bounding-box IoU (threshold 0.3), with unmatched target instances treated as additions and unmatched source instances treated as removals. The pipeline additionally identifies unrequested cross-category replacements when an added instance substantially overlaps a removed instance of another category, unless the corresponding addition and removal are both requested by the ground truth.

   Count-based instructions are generated from the observed source and target counts. For spatial instruction types, the pipeline constructs a difference mask from the source and edited-image segmentations, marks its connected components on the relevant image, and uses the resulting annotated image to generate spatial and color-spatial descriptions.

2. **Step 2: LLM-as-a-Judge Instruction Matching (`step_2_evaluate_predictions.py`).** The instruction describing the observed edit is compared with the corresponding ground-truth instruction using a structured LLM-based evaluator. The evaluator decomposes both instructions into edit, color, and spatial components and returns their matched, missing, and extra components, together with full-instruction hard- and soft-match indicators. The returned component statistics are then used to compute Count Match and the component-level precision, recall, and F1 metrics.

Component Extraction and Evaluation Rules

The evaluation prompt decomposes each instruction into three granular component categories:

- **Edit Components**: Defined as `(Action, Total Count, Base Object)`. Color and spatial descriptors are ignored. Same action–object groups are merged by summing their quantities (e.g., *"add one red onion and one white onion"* → *`["add two onions"]`*).
- **Color Components**: Defined as `(Count, Color, Object)`. Spatial location is ignored. Soft color equivalences are allowed within the same color family (e.g., light blue ≈ blue, dark green ≈ green, black ≈ dark gray, white ≈ beige), while contradictory colors (e.g., yellow vs. orange) are penalized.
- **Spatial Components**: Defined as `(Count, Object, Location)`. Spatial location is preserved. Soft spatial equivalences match logically equivalent regions (e.g., top right-middle ≈ top right, bottom middle ≈ lower center, left side ≈ far left).
- **Synonym and Subtype Normalization**: Object category subtypes and visually equivalent synonyms are recognized as valid matches for the base object category (e.g., *boot*, *sneaker*, or *sandal* match the base category *shoe*).

Formal Metric Definitions

From the component statistics returned by the LLM evaluator, the evaluation script computes the following formal metrics:

- **Exact Edit Accuracy Match Rate (Count Match)**: A strict binary metric evaluated per sample, requiring $N = \texttt{num_edit_components} > 0$. It requires `object_match` = 100, exact matching of all ground-truth edit actions, base object categories, and object counts (`correct_edit_bindings` = N, `correct_edit_count_bindings` = N), zero missing edit components (`missing_edit_components` = 0), zero extra unrequested edit components (`extra_edit_components` = 0), and an exact generated component count (`num_gen_edit_components` = N). Mismatches in color or spatial position do not invalidate Count Match if base actions, objects, and counts are exact.
- **Exact Full-Instruction Hard Match**: Returned directly by the LLM evaluator as 100% if and only if the generated instruction matches the ground truth perfectly as complete edit units with exact action, object, count, strict color, and strict spatial positioning, with zero missing or extra components; 0% otherwise.
- **Exact Full-Instruction Soft Match**: Returned directly by the LLM evaluator as 100% if and only if the generated instruction matches the ground truth as complete edit units using soft color and spatial equivalences, with exact action, object, and count, and zero missing or extra components; 0% otherwise.
- **Recall, Precision, and F1 Scores**: Computed for Edit, Color, and Spatial component bindings:

$$\text{Recall}_{\text{binding}} = \frac{\text{correct bindings}}{N_{\text{GT}}} \times 100 \tag{1}$$

$$\text{Precision}_{\text{binding}} = \frac{\text{correct bindings}}{N_{\text{Gen}}} \times 100 \tag{2}$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

  Corresponding count-sensitive variants ($\text{Recall}_{\text{count}}$, $\text{Precision}_{\text{count}}$, F1_{count}) substitute count-sensitive binding matches for standard binding matches.
- **Extra Unrequested Edit Component Rate**: The percentage of evaluated samples exhibiting unrequested modifications, defined as either count alterations on non-target background categories ($\Delta\text{count}_{\text{bg}} \neq 0$) or spurious opposite-action target operations (Figure 27).
- **Opposite-Action Error Rate (Add $\leftrightarrow$ Remove)**: The percentage of samples in which the model inverts the requested operation on a target object category (e.g., adding instances when instructed to remove, or removing instances when instructed to add) (Figure 29).
- **Incomplete Add-and-Remove Rate**: The percentage of compound `remove+add` instructions where the model executes only one of the two requested actions (Figure 30).
- **No-Edit Failure Rate**: The percentage of samples where the model produces zero detected instance additions or removals across all scene categories (Figure 26).

We compute these scores independently for each applicable sample and report their macro average across the dataset. Samples for which a metric has a zero denominator (e.g., no generated components for precision, or no ground-truth components for recall) are excluded from the corresponding macro average.

Qualitative Comparison with Lexical Instruction Metrics

Our evaluation uses a structured LLM-based comparison between the inferred and ground-truth edit instructions rather than relying directly on lexical similarity. This distinction is particularly relevant for MOMI, where instructions may contain multiple edit components together with color and spatial attributes. Two instructions can share much of their vocabulary while differing in a critical count, action, or attribute binding; conversely, semantically equivalent instructions may differ substantially in word order.

Table 30: **Qualitative comparison of lexical similarity and structured instruction evaluation on MOMI.** Representative stored production outputs illustrate limitations of lexical-overlap metrics for multi-component editing instructions. Substantial lexical similarity can persist despite incorrect counts, missing components, or incorrect color and spatial bindings. Conversely, sequence-sensitive metrics can assign substantially lower scores to semantically equivalent instructions whose clauses are reordered. The ground-truth and inferred instructions are shown verbatim.

Case	Ground-Truth Instruction	Inferred Instruction	Structured Outcome	Lexical Scores
Color-Detailed-Spatial	Add black and white mice at very top right, white, grey, and red mice just below them in upper right, black, green, and yellow mice in middle right area, and two black mice and one purple mouse at very bottom right.	Add black and white mice in far upper right corner, green mouse in center right area, black and yellow mice in lower right section, and three black mice and one purple mouse clustered near very bottom right edge.	**FAIL** 5/7 color and 5/10 spatial bindings matched.	BLEU-4: 0.321 METEOR: 0.594 ROUGE-L: 0.608
Detailed-Spatial	Add two socks in far bottom left corner, two in lower left-center area, two in lower right-center section, and one near far bottom right edge and remove shoe in top row, fourth from left.	Add two socks in bottom left corner, two in lower left-center, two in lower right-center, and one in bottom right corner, and remove two shoes from top row's center.	**FAIL** Wrong shoe count and location; 4/5 spatial bindings matched.	BLEU-4: 0.362 METEOR: 0.668 ROUGE-L: 0.762
Compound Add+Remove	Add seven black goats and three white goats and remove three blue trashcans, two red trashcans, and three yellow trashcans.	Add seven black goats and six white goats, and remove two red trashcans and four yellow trashcans.	**FAIL** Blue-trashcan component missing and multiple counts incorrect.	BLEU-4: 0.447 METEOR: 0.754 ROUGE-L: 0.811
Count Mismatch	Add two blue cushions, two red cushions, and five yellow cushions.	Add two blue cushions, two red cushions, and nine yellow cushions.	**FAIL** Yellow-cushion count is 9 rather than 5.	BLEU-4: 0.742 METEOR: 0.906 ROUGE-L: 0.909
Missing Component	Add seven buttons and remove seven coconuts.	Remove seven coconuts.	**FAIL** The add-buttons component is completely omitted.	BLEU-4: 0.148 METEOR: 0.446 ROUGE-L: 0.600
Semantic Reordering	Add one red trashcan and two yellow trashcans.	Add two yellow trashcans and one red trashcan.	**PASS** Exact semantic equivalence; Hard Match $= 100$.	BLEU-4: 0.248 METEOR: 0.938 ROUGE-L: 0.500

Table 30 reports these lexical scores together with the outcome of our structured evaluator. This analysis is qualitative and is not intended as a human-validated comparison of evaluator accuracy.

Table 30 illustrates two complementary limitations of lexical similarity for evaluating structured edit instructions. First, relatively high lexical similarity can persist despite substantive errors in the requested edit. In the count-mismatch example, the two instructions differ only in the requested number of yellow cushions, changing from five to nine, yet METEOR and ROUGE-L remain 0.906 and 0.909, respectively. Similarly, the compound multi-category example retains a ROUGE-L score of 0.811 and a METEOR score of 0.754 despite completely omitting the blue-trashcan component

and producing incorrect quantities for both white goats and yellow trashcans. The Detailed-Spatial example likewise obtains a ROUGE-L score of 0.762 despite an incorrect shoe count and spatial location. These cases show that lexical overlap alone does not explicitly verify whether counts and attributes remain correctly bound to their corresponding objects and edit operations.

Second, lexical similarity may decrease even when the structured edit semantics are unchanged. In the final example, the inferred instruction contains exactly the same requested additions as the ground truth but reverses the order of the two clauses. The structured evaluator therefore produces an exact hard match, while BLEU-4 decreases to 0.248 and ROUGE-L to 0.500. ROUGE-1 remains 1.0 and METEOR remains high at 0.938, illustrating that the sensitivity varies across lexical metrics.

These examples motivate the component-based comparison used in our evaluation. Rather than assigning a single similarity score to the complete instruction, our evaluator explicitly decomposes the two instructions into edit $(\text{action}, \text{count}, \text{object})$, color $(\text{count}, \text{color}, \text{object})$, and spatial $(\text{count}, \text{object}, \text{location})$ components and evaluates their bindings, counts, and missing or extra components. This structured representation is particularly important for the Color-Spatial, Detailed-Spatial, and Color-Detailed-Spatial instructions in MOMI, where a high degree of lexical overlap does not necessarily imply that the requested attributes are attached to the correct object instances.

Full Pipeline and Evaluation Prompt Specifications

Spatial Instruction Generation Prompt Below is the prompt specification (`generate_spatial_instructions.txt`) passed to Gemini 3.1 Pro along with the annotated source image (overlaid with SAM 3 target instance bounding boxes) to generate coarse and detailed spatial descriptors across four distinct instruction variants (Figure 52).

```
You are an expert multimodal visual instruction generator. Analyze the
provided image overlaid with bounding box region highlights corresponding to
the requested edit: "{instruction}".

Generate four distinct versions of the edit instruction based on the spatial
positions and colors of the highlighted regions:

Version 1: (Coarse Spatial Instruction)
- Describe object locations using coarse regional descriptors (e.g., "on left",
  "in center", "at bottom"), without including color attributes.

Version 2: (Coarse Color + Spatial Instruction)
- Combine instance-level object colors with coarse regional location
  descriptors (e.g., "two silver knives on left and one black knife in middle").

Version 3: (Detailed Spatial Instruction)
- Describe object locations using fine-grained scene-relative spatial
  descriptors (e.g., "near top left corner", "positioned along lower edge"),
  without including color attributes.

Version 4: (Color + Detailed Spatial Instruction)
- Combine instance-level object colors with fine-grained scene-relative spatial
  descriptors (e.g., "two silver knives near top left corner and one black knife
  along lower edge").

Format your output clearly with markers for "Version 1:", "Version 2:",
"Version 3:", and "Version 4:".
```

Figure 52: Prompt template for extracting coarse and detailed spatial instruction variants from annotated images (`generate_spatial_instructions.txt`).

LLM-as-a-Judge Instruction Matching Prompt (v2) Below is the full prompt template (`evaluate_instruction_match_prompt_v2.txt`) used in Step 2 for structured instruction evaluation and component extraction (Figures 53, 54, and 55).

Image Generation Prompt Specifications

Below are the prompt templates used by Gemini 2.5 Pro during the image generation phase to expand concise input captions into color-diverse descriptive prompts.

```
You are an expert evaluator of image-editing instructions.

Compare the Ground Truth instruction and Generated instruction.

Ground Truth: {0}
Generated: {1}

Return ONLY a raw JSON object. Do not use markdown, code fences, explanations, or extra text.
The response must start with "{" and end with "}".

Return EXACTLY these fields:

{
"object_match": 0,

"gt_summed_edit_components": [],
"gen_summed_edit_components": [],
"num_edit_components": 0,
"correct_edit_bindings": 0,
"correct_edit_count_bindings": 0,
"missing_edit_components": 0,
"extra_edit_components": 0,

"gt_summed_color_components": [],
"gen_summed_color_components": [],
"num_color_components": 0,
"correct_color_bindings": 0,
"correct_color_count_bindings": 0,
"missing_color_components": 0,
"extra_color_components": 0,

"gt_summed_spatial_components": [],
"gen_summed_spatial_components": [],
"num_spatial_components": 0,
"correct_spatial_bindings": 0,
"correct_spatial_count_bindings": 0,
"missing_spatial_components": 0,
"extra_spatial_components": 0,

"full_instruction_hard_match": 0,
"full_instruction_soft_match": 0
}

Definitions:

"object_match": 100 if all required base object types appear correctly, 50 if only some appear,
                0 if mostly missing/wrong. Ignore color and spatial attributes.

For all component extraction and matching, treat clear object subtypes, synonyms, or visually
equivalent category names as the GT base object when they reasonably refer to the same allowed
object type; for example, if GT requires "shoe", then boot/sneaker/sandal/slip-on in Generated
may be treated as shoe.

Edit components:

* A summed edit component is action + total count + base object type.
* Ignore color and spatial attributes.
* Merge repeated same action-object groups by summing counts.
* Example: "add one red onion and one white onion" -> ["add two onions"].
* Example: "add brown egg on middle left and speckled green egg on bottom left" -> ["add two eggs"].
* Example: "add five spoons and remove seven bowls" -> ["add five spoons", "remove seven bowls"].

Edit metrics:

* Compare gt_summed_edit_components to gen_summed_edit_components.
* num_edit_components = len(gt_summed_edit_components).
* correct_edit_bindings: GT edit components with matching action + base object, ignoring count.
* correct_edit_count_bindings: GT edit components with matching action + base object + exact count.
* missing_edit_components: GT edit components not matched by generated edit components.
* extra_edit_components: generated edit components not required by GT.
* If a generated edit component matches the GT action + base object but has the wrong count,
  count it as a correct_edit_binding, not a correct_edit_count_binding, and do not also count it as missing or extra
```

Figure 53: Prompt template for Step 2 LLM-as-a-Judge instruction evaluation and component extraction (Part 1/3: Schema, Object, and Edit definitions).

```
Color components:

* A summed color component is count + color + object.
* Ignore spatial location.
* Merge repeated same color-object groups by summing counts.
* Merge soft-equivalent shade variants of the same base color and object when they refer to the same required
  color family, e.g. "one bright brown coconut" + "three light brown coconuts" -> ["four brown coconuts"].
* Example: "one red tomato in top left and two red tomatoes in bottom right" -> ["three red tomatoes"].

Color metrics:

* Compare gt_summed_color_components to gen_summed_color_components.
* num_color_components = len(gt_summed_color_components).
* correct_color_bindings: GT color components with matching color + object, ignoring count.
* correct_color_count_bindings: GT color components with matching color + object + exact count.
* missing_color_components: GT color components not matched by generated color components.
* extra_color_components: generated color components not required by GT.
* Use soft color equivalence for bindings: light blue~=blue, dark green~=green, black~=dark gray, white~=beige.
  Do not match contradictory colors such as yellow vs orange or purple vs pink.
* If a generated color component matches the GT color + object but has the wrong count, count it as a
  correct_color_binding, not a correct_color_count_binding, and do not also count it as missing or extra.

Spatial components:

* A summed spatial component is count + object + location.
* Include color in the spatial component only when color is needed to distinguish object groups.
* Do NOT merge across different spatial locations.
* Merge only repeated same or soft-equivalent object-location groups.
* Example: "one red tomato in top left and two red tomatoes in bottom right" -> ["one red tomato in top left",
           "two red tomatoes in bottom right"].
* Example: "one apple in top left and two apples in upper left" -> ["three apples in top left/upper left"]
           if locations are soft-equivalent.

Spatial metrics:

* Compare gt_summed_spatial_components to gen_summed_spatial_components.
* num_spatial_components = len(gt_summed_spatial_components).
* correct_spatial_bindings: GT spatial components with matching object + location, ignoring count.
* correct_spatial_count_bindings: GT spatial components with matching object + location + exact count.
* missing_spatial_components: GT spatial components not matched by generated spatial components.
* extra_spatial_components: generated spatial components not required by GT.
* Use soft spatial equivalence for bindings: top right-middle~=top right, bottom middle~=lower center,
  upper middle~=top center, left side~=far left. Do not match contradictory locations such as top left vs bottom le
* If a generated spatial component matches the GT object + location but has the wrong count, count it as a
  correct_spatial_binding, not a correct_spatial_count_binding, and do not also count it as missing or extra.
```

Figure 54: Prompt template for Step 2 LLM-as-a-Judge instruction evaluation and component extraction (Part 2/3: Color and Spatial component rules).

```
Example:

Ground Truth: Add two blue bags on the left and one green table on the right.
Generated: Add two green bags on the right and one blue table on the left.

Expected key output:
{
"gt_summed_edit_components": ["add two bags", "add one table"],
"gen_summed_edit_components": ["add two bags", "add one table"],
"num_edit_components": 2,
"correct_edit_bindings": 2,
"correct_edit_count_bindings": 2,
"missing_edit_components": 0,
"extra_edit_components": 0,

"gt_summed_color_components": ["two blue bags", "one green table"],
"gen_summed_color_components": ["two green bags", "one blue table"],
"num_color_components": 2,
"correct_color_bindings": 0,
"correct_color_count_bindings": 0,
"missing_color_components": 2,
"extra_color_components": 2,

"gt_summed_spatial_components": ["two blue bags on the left", "one green table on the right"],
"gen_summed_spatial_components": ["two green bags on the right", "one blue table on the left"],
"num_spatial_components": 2,
"correct_spatial_bindings": 0,
"correct_spatial_count_bindings": 0,
"missing_spatial_components": 2,
"extra_spatial_components": 2,

"full_instruction_hard_match": 0,
"full_instruction_soft_match": 0
}

Reason: action, base objects, and counts are correct, but color and spatial bindings are attached to the wrong obje

Full-instruction metrics:

* full_instruction_hard_match = 100 only if the generated instruction exactly matches the GT as complete edit units
  same action, object, count, exact color, exact spatial location, and no missing/extra/wrong bindings. Otherwise 0
* full_instruction_soft_match = 100 only if the generated instruction matches the GT as complete edit units using
  soft color/spatial equivalence, with exact action/object/count and no missing/extra/wrong bindings. Otherwise 0.
```

Figure 55: Prompt template for Step 2 LLM-as-a-Judge instruction evaluation and component extraction (Part 3/3: Evaluation example, expected output, and full-instruction metrics).

Base Color Expansion Prompt The prompt template (`color_expand_image_caption.txt`) used to generate the initial expanded prompt ($i = 0$) with controlled color attributes across instances is shown in Figure 56.

```
Given a concise description of an image, extend it into a short prompt suitable for image generation, ensuring all key
Use between two and five distinct colors for the input objects only, while keeping the number of objects identical to t

Example 1
Input: "seven cats"
Output: "Seven black, white, gray, and orange cats sitting on a grassy meadow under a blue sky."

Example 2
Input: "eight eagles"
Output: "Eight brown and white eagles flying in the sky above mountains."

Example 3
Input: "3 red cats and two red koalas"
Output: "Three red, orange, dark brown, and light gray cats near two red koalas."

Your turn
Input: "{}"
```

Figure 56: Prompt template for initial color-diverse prompt expansion (`color_expand_image_caption.txt`).

Color Diversity Expansion Prompt The prompt template (`color_diversity_caption_prompt.txt`) used to synthesize subsequent diverse candidate prompts ($i > 0$) for the same input specification is shown in Figure 57.

```
Expand a brief image description into a short, concise, and realistic prompt for image generation.
Ensure the output includes clear visual details and avoids abstract or artistic interpretations, while differing signif
Describe the objects as clearly separated and individually visible, avoiding any wording that suggests a pile, cluster,
Use between two and five colors for the input objects only, while keeping the number of objects identical to the input.

Previously Generated Prompts:
{}

Input: {}
Output:
```

Figure 57: Prompt template for generating diverse color-expanded prompt candidates (`color_diversity_caption_prompt.txt`).

Source-Image Quality Filtering

Before verified generated images are used as editing sources, we apply two additional quality filters. First, we discard images containing target-object masks that touch the image boundary, which typically indicates truncated or partially visible object instances. Second, we discard images containing very small fragmented masks, which indicate poorly resolved or unreliable instance segmentations.

SAM 3 Instance Extraction and Post-Processing

We apply several filtering steps to obtain reliable instance-level predictions from SAM 3. We use a model confidence threshold of 0.4 and retain extracted instances only when their segmentation score is at least 0.3. We additionally deduplicate predictions using a mask-IoU threshold of 0.8, suppressing highly overlapping masks that correspond to the same physical object.

SAM 3 may also produce larger group-level predictions that cover multiple individual instances. Such predictions can lead to over-counting or inappropriate removal masks. We therefore suppress containing masks and bounding boxes before using SAM 3 predictions for downstream processing. A bounding box is discarded when it contains at least four smaller boxes with at least 80% coverage, while a mask is discarded when it contains at least two smaller masks with at least 50% coverage. Only predictions retained by both filters are used.

This processing is applied whenever SAM 3 predictions are used for source-image count verification, removal-mask selection, post-edit remnant detection, and benchmark evaluation.

Edit Verification and Count-Offset Calibration

During edit verification, we additionally account for systematic errors in the estimated source-image count. If SAM 3 and Gemini both produce counts that differ from the nominal expected counts by the same constant offset Δ across at least three removal edits derived from the same source image, we treat this agreement as evidence of a systematic source-count offset. We then recalibrate the expected counts using this offset and recover the corresponding verified edit family.

.8 Supervision and Fine-Tuning Details

We fine-tune the `Flux Kontext` base model using Low-Rank Adaptation (LoRA) on the $20,000$ verified edit pairs from the MOMI training set. The model is trained using the native `Flux Kontext` image-to-image diffusion training framework without passing edit masks as model inputs during fine-tuning or inference; conditioning relies strictly on the source image and textual edit instruction.

Hyperparameters and Training Configuration

- **Adaptation Architecture:** Low-Rank Adaptation (LoRA) applied to the diffusion transformer backbone with rank $r = 10$ and scaling factor $\alpha = 10$.
- **Image Resolution:** Native 1024×1024 pixels.
- **Batch Size & Compute:** Training was conducted across 6 NVIDIA H200 GPUs with a per-GPU batch size of 16 and gradient accumulation steps of 1, yielding an effective total batch size of 96.
- **Optimizer & Schedule:** Fine-tuned for 10 epochs using the AdamW optimizer with a fixed learning rate of 2×10^{-4}. Total training execution spanned approximately 2 days (~ 48 hours).

.9 Dataset Partitioning and Lineage Leakage Prevention

To prevent train–test leakage, partitioning across the MOMI, SOMI, and SOMI-Real splits is enforced at the *root image lineage level*. Specifically:

1. **Root Prompt Partitioning:** Base captions and generation prompts are partitioned prior to image synthesis. No prompt or semantic variant used for evaluation benchmarks is present in the training prompt set.
2. **Image Pair Lineage:** For every removal edit, the source image, the inpainted target image, and its reversed addition counterpart belong strictly to the same partition. Reversed addition pairs are never assigned to a different split than their corresponding removal source.
3. **Compound Edit Chains:** Compound edits (*Multi-Object Same-Action* and *Combined Remove+Add*) are composed exclusively from base images and single-object edits within the same partition.

This protocol guarantees that no endpoint image, background scene, or reversed edit pair appears across both training and evaluation splits.

.10 Paired Statistical Significance Analysis

To assess the robustness of the higher aggregate MOMI performance achieved by PGAMIE, we conduct sample-level paired comparisons between PGAMIE (16.33%, $49/300$) and the strongest baseline, Nanobanana Pro (14.33%, $43/300$).

McNemar Test and Contingency Matrix. Across the 300 test samples, both models fail on 223 samples and succeed on 15 samples. Among the discordant outcomes, PGAMIE succeeds while Nanobanana Pro fails on 34 samples ($b = 34$), whereas Nanobanana Pro succeeds while PGAMIE fails on 28 samples ($c = 28$). A continuity-corrected McNemar test yields $\chi^2 = 0.4032$ with $p = 0.5254$, while the corresponding exact two-sided binomial test yields $p = 0.5258$. Thus, the paired outcomes are consistent with the higher aggregate Edit Accuracy achieved by PGAMIE, which succeeds on more discordant samples than Nanobanana Pro.

Paired Bootstrap Confidence Interval. Using $10{,}000$ paired bootstrap resamples, the estimated accuracy difference between PGAMIE and Nanobanana Pro is $\Delta = +2.00$ percentage points, with a 95% confidence interval of $[-3.33\%, +7.33\%]$. The interval includes zero, indicating uncertainty in the magnitude of the aggregate advantage at the current 300-sample evaluation size. Nevertheless, the positive point estimate and paired outcome counts both favor PGAMIE, which also achieves a substantially larger numerical advantage on specific edit types, including *Multi-Object Same-Action* edits, where it achieves 11.0% compared with 4.6% for Nanobanana Pro.