VEGA: SPARSE MARK EXPERTS FOR COMPOSABLE VISUAL-INSTRUCTION EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Precise image editing has long sought spatial instructions that express not only what to change, but also where to change. Modern multimodal editors are strong at understanding semantic intent from language, yet precise spatial grounding remains difficult to express through text alone. Visual instructions, such as boxes, arrows, and doodles, provide this missing grounding directly in image coordinates, while existing editors struggle to execute such instructions reliably. We study this failure and find that visual marks are not simply inaccessible to the editor. A frozen editor can localize and manipulate these marks when they are treated as image content, yet often fails when the same marks serve as editing instructions. This gap suggests that existing editors have not learned to reliably use visual marks as editing controls. We propose VEGA, a lightweight adapter that turns visual instructions into executable editing controls. For each type of visual instruction, VEGA trains a specialized expert using flow matching with auxiliary attention supervision, encouraging the expert to route the corresponding spatial evidence into localized editing responses. Compound training further encourages each expert to remain selective in the presence of competing annotations. We then introduce *Attention Response Routing*, which uses these learned responses as token-wise responsibilities to compose independently trained experts along a shared generation trajectory, without parameter merging or joint expert training. Experiments on internal and external benchmarks show that VEGA is effective across multimodal editing models, substantially improving spatially grounded editing and enabling efficient composition of multiple visual instructions while preserving the original generation quality.

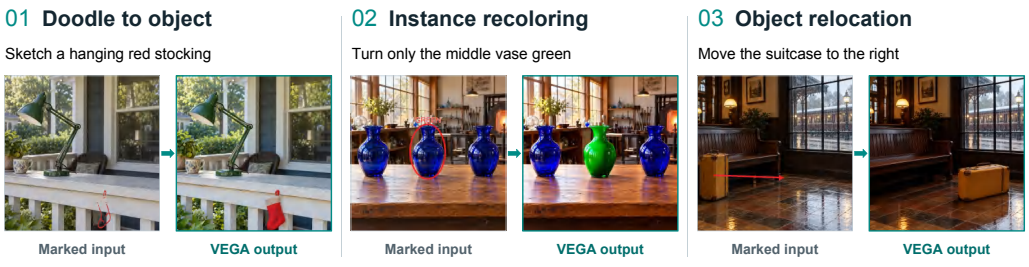

Figure 1: Visual instructions execute directly in the image. Three marked inputs request turning a stocking doodle into a hanging red stocking, recoloring only the middle vase, and moving a suitcase. **VEGA** executes each edit while removing the authored marks; no intermediate caption or structured plan is produced.

1 INTRODUCTION

Text-guided image editing has made substantial progress with diffusion-based generative models. Modern multimodal editors can perform increasingly complex changes in appearance and content through natural-language instructions Wu et al. (2025); Team et al. (2025). Despite these advances, precise spatial control remains challenging. Language effectively specifies *what* should change, but expressing exactly *where* the change should occur often requires indirect descriptions of the

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.



Model	Before	Seen (300)	Unseen (1K)
Qwen-Edit	0.12	0.91	0.71
LongCat-Edit	0.08	0.95	0.54

Figure 2: From visual access to execution. Left: a frozen editor can manipulate an authored mark as image content and perform the corresponding edit from explicit text, yet often fails to execute the instruction directly from the annotated image. **Right:** lightweight post-training connects visual instructions to editing behavior and transfers beyond the expressions observed during training. Seen and Unseen contain 300 and 1,000 evaluation examples, respectively.

image content. This gap motivates visual instructions that anchor editing intent directly to its spatial location in the image.

Visual instructions express this correspondence through annotations: a box anchors a request to a region, an arrow specifies a displacement, and text or a doodle places an intended change at its corresponding location. Because these annotations share the image plane with the content being edited, their relative spatial relationships are directly represented in the image. Existing systems typically handle such spatial instructions either by using a vision–language model to translate them into textual descriptions, or by introducing dedicated architectures and representations for spatial control Zhang et al. (2023); Li et al. (2023); Shi et al. (2024); Fu et al. (2024). These approaches are effective, but raise a more basic question: can the visual-conditioning pathway already present in modern multimodal editors itself support spatially grounded editing?

We investigate this question through a paired diagnostic (Fig. 2, left). A frozen editor can localize and remove an authored mark as image content, and can perform the corresponding transformation when the same request is explicitly given in text. Yet it often fails to execute the instruction conveyed by the annotated image itself. We call this failure mode *Reading without Routing*. The model can read the visual evidence and possesses the corresponding editing capability, but fails to reliably route that evidence into the localized editing action.

To determine whether this gap can be bridged through lightweight adaptation, we conduct a preliminary study on two multimodal editing backbones, Qwen-Edit and LongCat-Edit. Post-training on a restricted vocabulary of rendered expressions substantially improves direct execution on seen expressions and, importantly, transfers to unseen expressions (Fig. 2, right). This transfer indicates that adaptation does more than memorize a small collection of rendered commands: the native visual condition can be connected to editing behavior that generalizes beyond the expressions observed during training. These findings motivate treating visual annotations themselves as executable editing conditions.

Based on these observations, we introduce **VEGA** (**V**isual **E**xecutable **G**uidance **A**daptation), a lightweight framework for executing visual instructions through the editor's native conditioning pathway. For each visual-instruction family, we train an independent sparse expert on a frozen editor. Flow-matching supervision learns the corresponding editing behavior, while routing supervision aligns the expert's visual response with the spatial support of its instruction. We further train each expert to remain selective in the presence of other annotations: it executes its own instruction while preserving unrelated instructions and their associated content. The resulting experts therefore learn both localized execution and robust response boundaries when multiple visual instructions coexist.

This sparse and selective behavior makes independently learned visual controls naturally composable. Rather than globally merging expert parameters or independently generating and mixing multiple predictions, we introduce *Attention Response Routing* (ARR) to compose expert responses within a shared denoising trajectory. ARR uses the attention responses of the selected experts as spatial responsibilities, allowing their contributions to vary across image locations. Each expert can therefore act where its visual instruction is relevant while multiple experts jointly refine the same image. This enables flexible composition of independently trained visual controls without joint multi-expert training or inference-time edit masks.

We evaluate **VEGA** on our All-in-Image-Bench and on the external VP-Bench Yi et al. (2026). Across text, handwriting, boxes, masks, doodles, and directional controls, **VEGA** improves visual instruction execution and spatial localization over existing editors and visual-control baselines. Controlled ablations further show that routing supervision substantially improves localization and reduces wrong-region responses, while selective expert training preserves individual capabilities un-

der competing annotations. For composed instructions, ARR outperforms parameter merging and provides higher edit localization precision than both baselines, while retaining inference cost close to a single native edit.

Our contributions are threefold:

- We identify *Reading without Routing*, a failure mode in which multimodal editors can perceive visual annotations and perform the corresponding edits, yet fail to route the visual evidence into localized editing actions. Paired diagnostics and lightweight adaptation establish this gap and its learnability.
- We introduce **VEGA**, which learns localized and selective experts for different visual-instruction families, together with ARR, which composes their responses spatially within a shared generation trajectory.
- We construct All-in-Image-45K, a 44,981-example visual-instruction editing dataset, and All-in-Image-Bench, a 600-example held-out benchmark spanning diverse visual instructions and their compositions, and validate **VEGA** on both our benchmark and the external VP-Bench.

2 Related Work

Text-Conditioned Diffusion Editing. Early diffusion editing methods manipulate attention or internal features to change image content while preserving structure Hertz et al. (2022); Cao et al. (2023). InstructPix2Pix established learning edits from synthetic instruction–image pairs Brooks et al. (2023), subsequently extended through richer supervision and multimodal understanding Sheynin et al. (2024); Fu et al. (2024); Geng et al. (2024). However, language often underspecifies spatial reference and multi-object grounding, and stronger recognition does not guarantee executable visual control. **VEGA** studies how instructions drawn directly on an image can guide localized editing through a pretrained editor.

Structured Visual Control for Diffusion Models. ControlNet and T2I-Adapter inject spatial conditions through side networks or lightweight adapters Zhang et al. (2023); Mou et al. (2024). GLIGEN grounds generation with boxes and phrases Li et al. (2023), while IP-Adapter and drag-based methods provide reference-image and point-level control Ye et al. (2023); Shi et al. (2024). These methods support predefined control formats through dedicated conditioning mechanisms. FlowInOne unifies heterogeneous requests as visual prompts for image-in, image-out generation Yi et al. (2026). **VEGA** learns localized and selective visual-control experts through lightweight adaptation of a frozen pretrained editor.

3 **VEGA**: Learning and Composing Visual-Instruction Experts

Given a source image I, the user specifies edits by placing visual instructions directly on the image, yielding an annotated image I_{mark}. **VEGA** takes I_{mark} as its editing condition, with a fixed, task-generic trigger activating visual-instruction editing. The requested transformations and their spatial locations are specified entirely by the annotations in the image. We denote the resulting editing condition by C, leaving the fixed trigger implicit in the notation. Each instruction family is handled by an independently trained expert. At inference, the experts selected for the requested families read the same annotated image, and Attention Response Routing (ARR) composes their contributions within a shared denoising process (Fig. 3).

3.1 Learning Sparse Visual-Instruction Experts

Different instruction families express different spatial relations: a box selects a region, an arrow specifies motion, and a doodle supplies a shape. We learn each convention with a separate LoRA expert F_m on the condition-side key and value projections Hu et al. (2021), keeping the pretrained encoder and backbone weights frozen. Each expert first learns the requested transformation and its spatial support, then learns to execute its own instructions selectively when other marks are present.

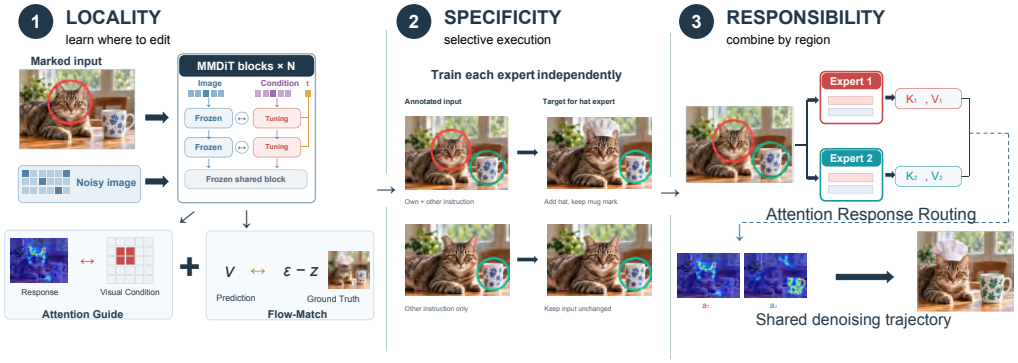

Figure 3: From localized response to compositional execution. Flow matching learns the requested transformation, while auxiliary attention supervision localizes the expert's visual response. Selective execution training teaches each expert to execute its own instructions while preserving unrelated instructions and their marks. At inference, ARR combines expert key–value representations through joint attention, whose spatially varying responses determine each expert's contribution along a shared generation trajectory. Images and response maps are schematic.

Learning the transformation. For target latent z, noise $\epsilon \sim \mathcal{N}(0, I)$, and noise level σ sampled from the training schedule, we construct $x_\sigma = (1-\sigma)z + \sigma\epsilon$ and target velocity $u_\sigma = \epsilon - z$. The standard flow-matching objective Lipman et al. (2022) is

$$\mathcal{L}^m_{\mathrm{FM}} = \mathbb{E}\Big[\|F_m(x_\sigma, \sigma; C) - u_\sigma\|_2^2\Big]. \tag{1}$$

Localizing the visual response. Target-image supervision leaves the correspondence between a visual instruction and its intended edit region implicit. Inspired by SWIM Sun et al. (2026), which improves object grounding through attention supervision, we supervise this correspondence within the editor's visual-conditioning pathway. Generated-image queries retrieve visual information through attention to condition keys; we encourage this response to concentrate on the intended edit region.

Let $\mathcal{P}$ index generated-image tokens, $\mathcal{K}$ all keys available to the attention layer, and $\mathcal{C}^v \subset \mathcal{K}$ the visual-condition key positions, shared across experts. For expert m, define

$$A^m_{\ell,h,p,k} = \mathrm{softmax}_{k\in\mathcal{K}}\left(\frac{q^{\mathrm{img}}_{\ell hp}(k^m_{\ell hk})^\top}{\sqrt{d}}\right), \tag{2}$$

where H denotes the number of attention heads and d the head dimension. The spatial response of expert m at image token p is the attention mass assigned to the visual-condition keys, averaged across heads:

$$a^m_{\ell p} = \frac{1}{H}\sum_{h=1}^{H}\sum_{k\in\mathcal{C}^v} A^m_{\ell,h,p,k}. \tag{3}$$

Thus, $a^m_{\ell p}$ measures how strongly generated location p retrieves the shared visual condition through expert m.

A training mask M^m marks the audited region where the edit should occur. We normalize this mask and the response over generated-image locations,

$$\widetilde{M}^m_p = \frac{M^m_p}{\sum_{j\in\mathcal{P}} M^m_j}, \qquad \widetilde{a}^m_{\ell p} = \frac{a^m_{\ell p}}{\sum_{j\in\mathcal{P}} a^m_{\ell j}}, \tag{4}$$

and minimize

$$\mathcal{L}^m_{\mathrm{route}} = -\frac{1}{|\mathcal{S}|}\sum_{\ell\in\mathcal{S}}\sum_{p\in\mathcal{P}} \widetilde{M}^m_p \log(\widetilde{a}^m_{\ell p} + \varepsilon), \tag{5}$$

over supervised layers $\mathcal{S}$. The single-control objective is

$$\mathcal{L}^m_{\mathrm{expert}} = \mathcal{L}^m_{\mathrm{FM}} + \lambda_{\mathrm{attn}}\mathcal{L}^m_{\mathrm{route}}. \tag{6}$$

The auxiliary term supplies a direct spatial training signal alongside the editing objective. The edit-region mask is not needed at inference.

Learning selective execution. We next train each expert to distinguish its own instructions from other marks in the same image. For expert m, a training input may contain only instructions from other families, or combine them with an instruction from family m. Its target image executes only the family-m instruction, removes the corresponding mark, and preserves the other instructions and their marks. When no family-m instruction is present, the target is the unchanged annotated input. For example, a recoloring expert leaves an image containing only a motion arrow unchanged. Given both a recoloring request and a motion arrow, it changes the requested color while preserving the object positions and the arrow.

Let Y_m denote this expert-specific target, z_m its latent, and C_{mark} the editing condition derived from the annotated input. We continue training each expert independently with flow matching:

$$\mathcal{L}^m_{\mathrm{sel}} = \mathbb{E}\Big[\|F_m(x^m_\sigma, \sigma; C_{\mathrm{mark}}) - (\epsilon - z_m)\|_2^2\Big], \qquad x^m_\sigma = (1-\sigma)z_m + \sigma\epsilon. \tag{7}$$

These targets teach an expert to execute its own request while leaving unrelated visual instructions available to the other experts.

3.2 Test-Time Composition with ARR

At inference, the selected experts read the same annotated image and share the evolving image tokens. ARR combines their key and value representations in a single attention operation, allowing each image query to retrieve information from the experts according to its spatial location.

Shared projections and expert key–value banks. For clarity, we omit the layer and head indices below. We compute the frozen base projections K_0, V_0 once and apply each selected expert's low-rank increments to obtain

$$K_m = K_0 + \Delta K_m, \qquad V_m = V_0 + \Delta V_m. \tag{8}$$

Here the expert increments act on the visual-condition positions; the remaining key and value projections are shared. We concatenate the M expert banks along the token dimension and evaluate joint attention:

$$\widehat{O} = \mathrm{softmax}\left(\frac{Q[K_1; \ldots; K_M]^\top}{\sqrt{d}}\right)[V_1; \ldots; V_M]. \tag{9}$$

The softmax is taken jointly over the concatenated keys. The base projections and image queries are computed once; the experts contribute through their additional low-rank projections.

Attention responses as spatial responsibilities. Joint attention also admits an exact interpretation as spatially weighted expert outputs. For image query q_p, define the score $s_{mpk} = q_p k_{mk}^\top/\sqrt{d}$ and the total unnormalized response of expert bank m as $Z_{mp} = \sum_k \exp(s_{mpk})$. The attention mass assigned to that bank is

$$w_{mp} = \frac{Z_{mp}}{\sum_{j=1}^M Z_{jp}}, \qquad \sum_{m=1}^M w_{mp} = 1. \tag{10}$$

Writing $O^m_p = \sum_k \exp(s_{mpk}) v_{mk}/Z_{mp}$ for attention normalized within expert bank m, Eq. 9 becomes

$$\widehat{O}_p = \sum_{m=1}^M w_{mp} O^m_p. \tag{11}$$

Thus, the attention response to each expert bank determines its contribution at each image location. These responsibilities arise directly from the joint softmax and vary across image queries and attention heads. Spatial supervision localizes the experts' visual responses, while selective execution training preserves their distinct behaviors under the shared multi-mark input.

The combined attention output returns to the shared backbone. The experts share the MLPs, residual stream, and denoising schedule, jointly refining one image throughout composition. For overlapping edits, we initialize the layout with individual experts during the first two denoising steps before applying shared composition (Appendix E.2). Algorithms 1 and 2 in Appendix A summarize training and inference.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Model and training. We instantiate **VEGA** on Qwen-Image-Edit-2509 Wu et al. (2025). Each instruction family uses a rank-64 LoRA expert with $\alpha = 128$ on the condition-side key and value projections; the pretrained encoder and backbone weights remain frozen. We first train each expert with flow matching and auxiliary attention supervision, then continue with selective execution training on expert-specific targets. Attention supervision uses layers 20, 40, and 59 with 64 sampled image queries. Its weight starts at $\lambda_{\text{attn}} = 0.02$ and is rescaled according to the FM loss. Inference uses 20 denoising steps and CFG 1. ARR shares the base projections and combines expert K/V banks through joint attention.

Data and inputs. All-in-Image-45K contains 44,981 audited source–mark–target training triples. All-in-Image-Bench contains 360 manually checked single-control examples and 240 composition examples, with no training overlap. The main comparison (Table 1) evaluates the single-control split, while the composition study (Table 4) evaluates the two-control split. Single-control results are averaged over three random seeds. Each composition example combines two expert families; qualitative results additionally show three simultaneous controls. **VEGA** uses the annotated image as its editing condition and a fixed, task-generic trigger to activate visual editing. The annotations specify the requested object, transformation, and spatial relation. Comparison editors receive the same annotated image with a concise explanation of the annotation convention and a request to execute it. Appendices B and C describe dataset construction and evaluation inputs.

Evaluation. Strict execution measures whether the requested edit is completed; a compound request succeeds only when every requested control is completed. CLIP-GT measures target-content fidelity, and Edit-IoU measures the overlap between actual image changes and the annotated edit regions. Outside-region PSNR and LPIPS measure preservation. We use the same target-blind Qwen3-VL-4B judge and spatial evaluators across methods; edit-region annotations are provided only to the evaluator. Marker removal and directional DINO for motion provide complementary diagnostics. Metric definitions appear in Appendix C. Human preference is evaluated on 100 sets of editing outputs, with one selected output per set (Appendix J).

4.2 MAIN RESULTS

Table 1 compares single-control editing on All-in-Image-Bench. **VEGA** achieves the highest strict execution score (84.35) and Edit-IoU (63.15), with a CLIP-GT score of 91.75. Its strict execution score exceeds the strongest comparison editor, Nano-banana (69.26), by 15.09 points. Relative to the frozen Qwen editor, it also improves outside-region PSNR from 16.18 to 17.83 and reduces outside-region LPIPS from 0.421 to 0.371. In the 100-set human preference study, **VEGA** receives 22% of the selections, ranking second to Nano-banana at 25% and ahead of the other compared editors. Figure 4 illustrates how these gains appear in individual edits: **VEGA** selects the annotated target and executes the requested transformation across the displayed visual instructions.

4.3 TRAINING SPARSE EXPERTS

Learning where to execute. We isolate attention supervision by holding initialization, training data, FM updates, and inference settings fixed. Adding the auxiliary objective raises edit success from 63.3% to 73.0% and Edit-IoU from 39.5% to 66.8% (Table 2). Wrong-instance edits decrease from 18.9% to 9.5%, while outside-region LPIPS decreases from 0.42 to 0.29. These results show that attention supervision improves execution, localization, and preservation outside the edit region.

Figure 5 connects this behavioral improvement to localized image-query-to-condition-key responses. Supervised responses align with the selected mug, directed path, and authored shape across layers. Held-out response IoU rises from .104 to .783 for box text and from .168 to .888 for doodles. Appendix D.1 provides all families.

Executing only the expert's own instruction. We evaluate selective execution training on 40 matched held-out compound examples, with 10 examples for each of the four control families (Ta-

Table 1: Single-control editing on All-in-Image-Bench. Automatic metrics on the 360-example single-control split are averaged over three random seeds. Best and second-best results are marked in **bold** and underline, respectively. **Strict Exec.** measures requested-edit execution, **CLIP-GT** measures target-content fidelity, and **Edit-IoU** measures the spatial localization of the actual change. **Outside PSNR** and **Outside LPIPS** measure preservation beyond the authorized edit region. **Human Pref.** reports the percentage of selections across 100 sets of editing outputs, with one selection per set.

Method	Strict Exec.↑	CLIP-GT↑	Edit-IoU↑	Outside PSNR↑	Outside LPIPS↓	Human Pref.↑
InstructPix2Pix	0.00	62.57	53.02	15.26	0.548	0.0
FLUX	15.00	72.16	44.15	15.98	0.445	9.0
LongCat-Image-Edit	28.33	75.12	38.83	**23.36**	**0.116**	16.0
Qwen-Image-Edit	20.10	74.34	50.17	16.18	0.421	14.0
OmniGen2	25.65	75.91	32.8	13.42	0.382	14.0
Nano-banana	69.26	**94.32**	37.6	14.86	0.136	**25.0**
VEGA (Qwen-Image-Edit)	**84.35**	91.75	**63.15**	17.83	0.371	22.0

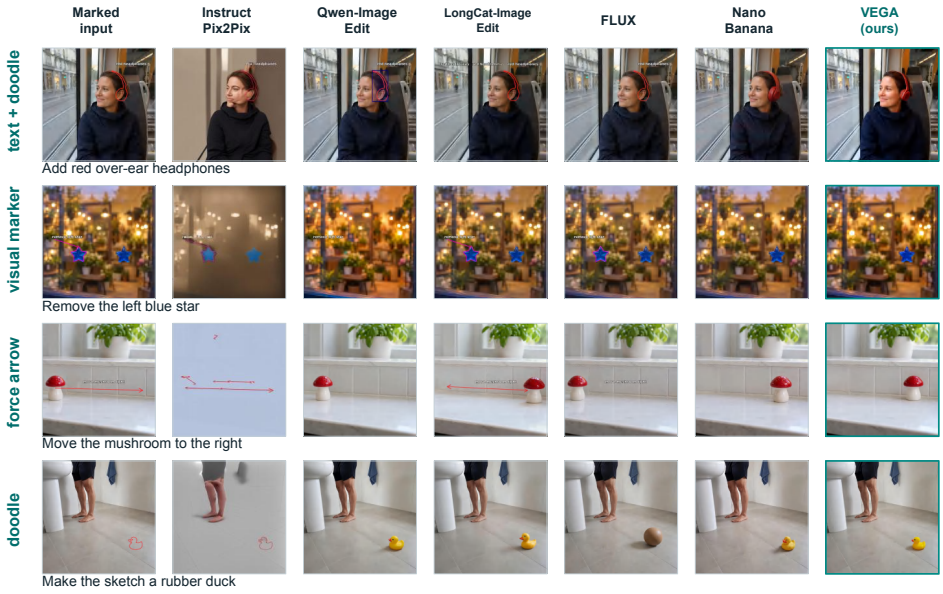

Figure 4: Qualitative comparison across visual instructions. Each row starts from the same marked image and compares outputs from existing editing models with **VEGA**. Row labels state the requested edit; the four rows cover text-guided doodle, visual-marker, force-arrow, and doodle controls.

Table 2: Matched attention-supervision ablation. Both rows use identical expert initialization, training data, FM schedules, and inference settings; only the auxiliary attention objective changes. Edit success, Edit-IoU, and wrong-object frequency are percentages. Outside LPIPS measures preservation on unaffected pixels (lower is better).

Expert objective	Edit succ.↑	Edit-IoU↑	Wrong obj.↓	Outside LPIPS↓
FM only	63.3	39.5	18.9	0.42
FM + Routing	**73.0**	**66.8**	**9.5**	**0.29**

ble 3). After this stage, own-task completion increases from 19/40 to 28/40, while false activation decreases from 8/40 to 2/40. Single-to-compound response agreement increases from .76 to .87, and response leakage into competing regions falls from 15.4% to 8.4%. These results show improved execution of the expert's assigned task in the presence of other instructions. Appendix E.1 reports per-family capability retention under composition.

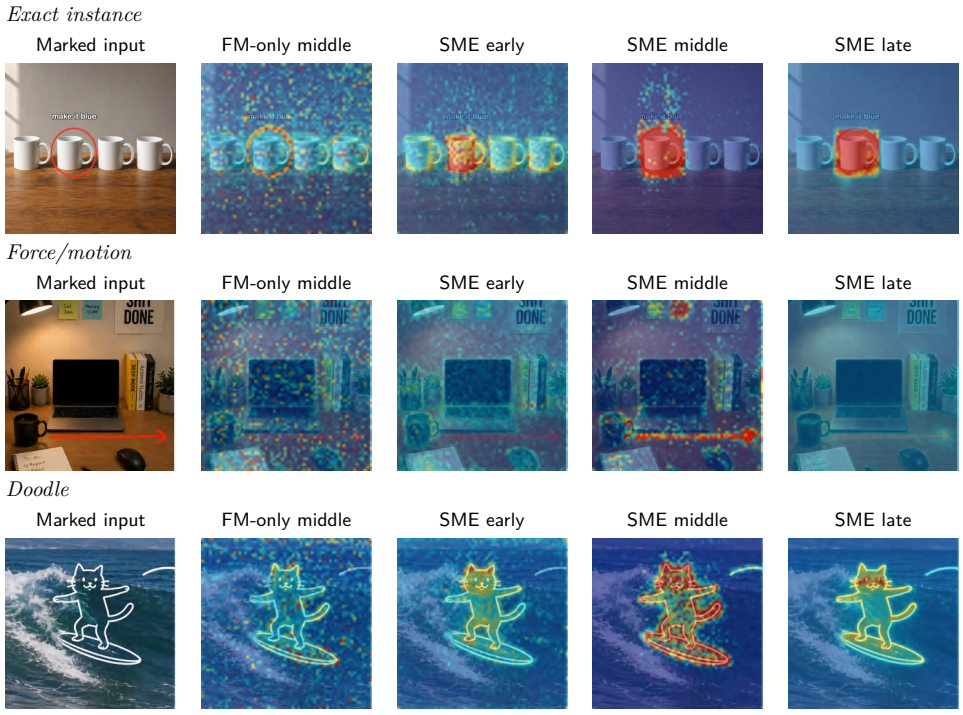

Figure 5: Routing localizes expert responses. Columns compare the annotated input, FM-only response, and supervised responses at early, middle, and late attention layers; rows test exact-instance selection, motion, and doodle geometry. SME denotes a routing-supervised sparse mark expert.

Table 3: Expert specificity under compound instructions. Both rows use the same compound inputs. Cosine measures single-to-compound response agreement; leakage is the percentage of response in competing regions. False activation and own-task success are counts over 40 examples, with 10 examples per control family.

Expert objective	Cos.↑	Leak↓	False act.↓	Own succ.↑
FM + Routing	.76	15.4	8/40	19/40
+ Selective execution	**.87**	**8.4**	**2/40**	**28/40**

4.4 Test-Time Composition with ARR

We study expert composition on the 240-example two-control split (Table 4). Parameter merging combines the expert weights before inference. Independent prediction mixing runs each of the two experts separately at every denoising step and averages their velocity predictions to update the shared latent. ARR instead combines expert K/V banks through joint attention, sharing the base projections and backbone computation within one denoising trajectory.

Independent prediction mixing achieves the highest strict all-control success at 57.7%, followed by ARR at 49.8% and parameter merging at 30.0%. ARR achieves the highest Edit-IoU at 71.0%, exceeding independent prediction mixing by 21.7 points. It also improves outside-region PSNR from 16.9 to 18.4 relative to independent prediction mixing; parameter merging yields the highest PSNR at 19.6. These results show a trade-off: independent prediction mixing completes more compound requests, while ARR provides more localized edits and better background preservation than that baseline through shared-backbone composition. Figure 6 shows multi-control editing results.

In the qualitative comparison, LoRA averaging and universal routing produce partial or interfering changes. The pigeon and bathtub scenes require semantic and relocation edits at distinct locations; ARR realizes both in one image. The final row extends this spatial coordination to three controls.

4.5 Generalization and Compatibility

VEGA generalizes across annotation styles and preserves editing compatibility. On 300 paired tasks, human-drawn instructions achieve .83 CLIP-GT versus .58 for the native editor, only .05 below reg-

Table 4: Comparison of expert composition methods. Results on the 240-example two-control split use the same trained experts, inputs, initial noise, and denoising schedule. Strict success requires both requested edits to be completed. Edit-IoU measures edit localization, PSNR measures preservation outside the edit region, and cost is relative to a single native edit. The independent-prediction baseline uses two separate full-model forwards per denoising step; its 2.00× entry denotes forward-pass count.

Composition	Output mixing	Spatial	Strict↑	Edit-IoU↑	PSNR↑	Cost↓
Parameter merge	–	–	30.0	35.6	**19.6**	**1.00**×
Independent prediction mixing	✓	–	**57.7**	49.3	16.9	2.00×
ARR	✓	✓	49.8	**71.0**	18.4	1.13×

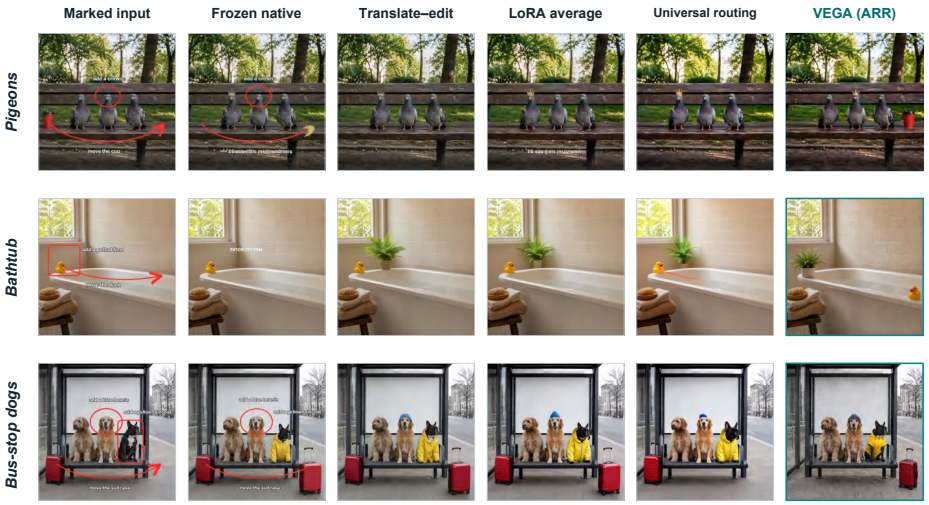

Figure 6: Spatial composition of specialized editing behaviors. Each marked image contains multiple simultaneous visual instructions. The frozen editor frequently leaves part of the requested edit unexecuted, while LoRA averaging and universal routing produce partial or interfering changes. ARR preserves the specialized behaviors and assigns them to their intended image regions within a single generation trajectory. The final row additionally illustrates composition of three simultaneous controls.

ular annotations. In repeated-object scenes, target accuracy and non-target preservation reach 92.4% and 94.5%. VEGA also preserves 97.0% of scene text while removing 98.0% of instruction overlays (Appendix H), and retains ordinary text-guided editing performance on MagicBrush (Appendix F). On VP-Bench, it achieves the highest spatial precision (3.51) and content consistency (4.29) among evaluated methods, with a four-dimensional average of 3.77 (Appendix I).

5 Conclusion

Visual-instruction failures are not always failures of perception. Our diagnostic, *Reading without Routing*, shows that an editor can perceive visual marks and perform the corresponding transformations, yet fail to execute the instructions conveyed by those marks. **VEGA** addresses this gap by learning lightweight experts that connect visual instructions to localized editing behavior. Attention supervision teaches each expert where to act, while selective execution training teaches it to preserve unrelated instructions. ARR then composes these independently trained experts through joint attention within a shared denoising trajectory. Experiments on All-in-Image-Bench and VP-Bench demonstrate improved visual-instruction execution and spatial precision, alongside efficient multi-control composition. These results suggest that adapting how a pretrained editor uses visual conditions can turn image-drawn annotations into precise, composable editing commands.

AI Use Statement

In this work, we used generative AI tools to assist with translation during manuscript drafting and to generate and filter synthetic training data. We have not used generative AI tools for developing

theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, cleaning or reformatting datasets, supporting qualitative or thematic data analysis, and interpreting results, and these tasks are not applicable to this work. Additionally, we used generative AI tools to edit the paper to improve readability. We have reviewed all AI-assisted work: all AI-assisted text was checked and revised by the authors for accuracy and clarity. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide extensive detail to support reproducibility of **VEGA**. The SPARSE MARK EXPERTS training objectives and the ARR composition rule are described at the implementation level in Sections 3.1 and 3.2, with the backbone, expert configuration, and training schedule given in Section 4.1. The construction of All-in-Image-45K and All-in-Image-Bench, along with the full evaluation protocol, metrics, and baseline input settings, is provided in Appendices B and C. Additional quantitative and qualitative results supporting the main-paper claims are reported in Appendices D.1 through H.2. We will publicly release **VEGA**, including All-in-Image-45K, All-in-Image-Bench with its region annotations, the training and evaluation codebase, and the trained expert checkpoints used to produce all reported results.

REFERENCES

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

Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22503–22513. IEEE, 2023.

Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Wang, Yinfei Yang, and Zhe Gan. Guiding instruction-based image editing via multimodal large language models. In *International Conference on Learning Representations*, volume 2024, pp. 54820–54833, 2024.

Zigang Geng, Binxin Yang, Tiankai Hang, Chen Li, Shuyang Gu, Ting Zhang, Jianmin Bao, Zheng Zhang, Houqiang Li, Han Hu, et al. Instructdiffusion: A generalist modeling interface for vision tasks. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12709–12720. IEEE, 2024.

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

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. Gligen: Open-set grounded text-to-image generation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22511–22521. IEEE, 2023.

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

Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 4296–4304, 2024.

Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu edit: Precise image editing via recognition and generation tasks. In *2024 ieee/cvf conference on computer vision and pattern recognition (cvpr)*, pp. 8871–8879. IEEE, 2024.

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

Boyuan Sun, Bo-Wen Yin, Yuan-Ming Li, Xihan Wei, and Qibin Hou. See what i mean: Aligning vision and language representations for video fine-grained object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36814–36827, 2026.

Meituan LongCat Team, Hanghang Ma, Haoxian Tan, Jiale Huang, Junqiang Wu, Jun-Yan He, Lishuai Gao, Songlin Xiao, Xiaoming Wei, Xiaoqi Ma, et al. Longcat-image technical report. *arXiv preprint arXiv:2512.07584*, 2025.

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

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

Junchao Yi, Rui Zhao, Jiahao Tang, Weixian Lei, Linjie Li, Qisheng Su, Zhengyuan Yang, Lijuan Wang, Xiaofeng Zhu, and Alex Jinpeng Wang. Flowinone:unifying multimodal generation as image-in, image-out flow matching, 2026. URL https://arxiv.org/abs/2604.06757.

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

APPENDIX

CONTENTS

A Training and Inference Algorithms . . . 14

B Data Engine and All-in-Image-Bench . . . 15

C Implementation and Evaluation Protocols . . . 16

C.1 Training Configuration . . . 16

C.2 Evaluation Inputs and Scoring . . . 17

C.3 Composed-Control Protocol . . . 17

D Attention Supervision and Response Diagnostics . . . 18

D.1 Family-Level Responses and Adaptation Scope . . . 19

E Composition Analysis . . . 20

E.1 Per-Expert Capability Retention . . . 20

E.2 Overlapping Edit Regions and Early Layout Initialization . . . 20

F Trigger Design and Text-Guided Editing Compatibility . . . 21

G Additional Qualitative Results . . . 21

H Interface Robustness and Spatial Correspondence . . . 25

H.1 Human-Drawn Instructions and Spatial Disambiguation . . . 25

H.2 Generalization to Unseen Visual-Text Instructions . . . 25

H.3 Natural Scene-Text Preservation and Authored-Mark Removal . . . 25

I Evaluation on VP-Bench . . . 27

J Human Preference Study . . . 27

A Training and Inference Algorithms

Algorithms 1 and 2 summarize expert training and test-time composition in Section 3. The first algorithm learns an expert's transformation, locality, and selective execution. The second combines the selected experts through joint attention on a shared generation trajectory.

Algorithm 1 Learning one sparse visual-instruction expert

Require: Instruction family m; single-mark source–target pairs with training edit-region mask M^m; marked inputs with expert-specific selective targets Y_m; frozen editor
Ensure: Specialized expert F_m
Initialize the condition-side K/V LoRA expert F_m.
Learn transformation and locality
for single-mark training minibatches **do**
    Form condition C_m from the annotated image and fixed trigger.
    Sample target latent z, noise ϵ, and level σ; set $x_\sigma \leftarrow (1-\sigma)z + \sigma\epsilon$ and $u_\sigma \leftarrow \epsilon - z$.
    Evaluate $F_m(x_\sigma, \sigma; C_m)$ and its image-query-to-condition-key response $a^m_{\ell p}$.
    Update F_m with $\mathcal{L}^m_{\mathrm{FM}} + \lambda_{\mathrm{attn}}\mathcal{L}^m_{\mathrm{route}}$ (Eqs. 1–6), using M^m only for routing supervision.
end for
Learn selective execution
for marked-input minibatches with selective targets Y_m **do**
    Form condition C_{mark} from the annotated input and fixed trigger.
    Set Y_m to execute only family-m instructions and preserve all other instructions and marks; use the unchanged marked input when family m is absent.
    Encode Y_m as z_m; sample ϵ, σ and set $x^m_\sigma \leftarrow (1-\sigma)z_m + \sigma\epsilon$.
    Update F_m with $\mathcal{L}^m_{\mathrm{sel}}$ (Eq. 7).
end for
return F_m

Algorithm 2 Compositional execution by Attention Response Routing

Require: Active expert families $\mathcal{M}$; their specialized experts $\{F_m : m \in \mathcal{M}\}$; one compound marked image I_{multi}; generic trigger; initial noisy state and denoising schedule
Ensure: Edited image
Form C_{multi} from the annotated image and fixed trigger; initialize one shared generated-image state.
if the requested edits overlap **then**
    Use two initial denoising steps to establish a shared layout from independent expert predictions (Appendix E.2).
end if
for each remaining denoising step **do**
    for each attention layer ℓ **do**
        Compute the shared queries Q and base projections K_0, V_0.
        for each active expert $m \in \mathcal{M}$ **do**
            Compute the condition-side LoRA increments $\Delta K_m, \Delta V_m$.
            Form $K_m \leftarrow K_0 + \Delta K_m$ and $V_m \leftarrow V_0 + \Delta V_m$.
        end for
        Concatenate expert key and value banks along the token dimension.
        Evaluate one joint attention operation (Eq. 9) and return its output to the shared backbone.
    end for
    Advance the single shared denoising trajectory.
end for
return the decoded edited image

All selected experts read the same compound condition. ARR uses neither expert-specific image crops nor edit-region masks at inference. For overlapping edits, the two layout steps precede joint attention; the total schedule remains 20 denoising steps.

B Data Engine and All-in-Image-Bench

Figure 7 illustrates the visual instruction forms and source-to-target changes represented by the data construction process.

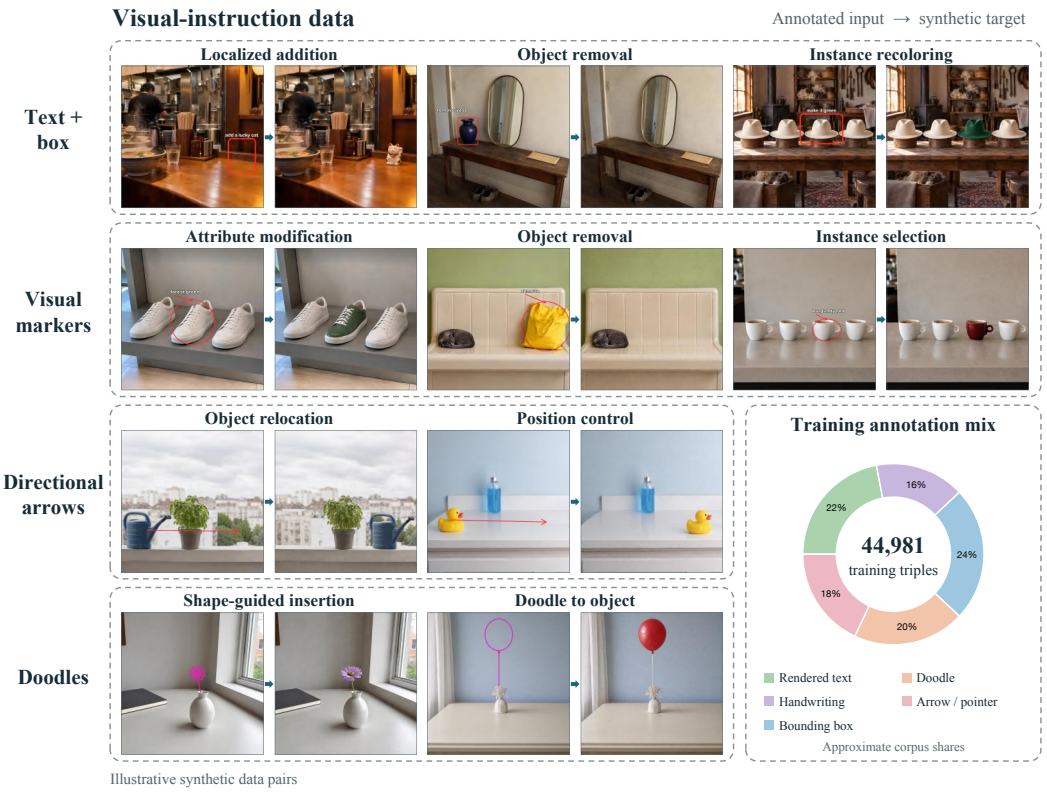

Figure 7: Visual-instruction data overview. Each pair shows an annotated input and its synthetic target, covering box-grounded addition, removal, and recoloring; marker-guided attribute changes and removal; arrow-directed relocation, and doodle-guided insertion. These are synthetic data illustrations. The chart summarizes the approximate training-corpus annotation proportions; annotation forms and expert families are distinct groupings.

Single-mark expert data. We construct source–marked–target triples in three steps. First, an instruction generator proposes local additions, removals, attribute edits, and spatial relocations; strong image editors synthesize the corresponding targets. Second, grounding models and VLM audits recover the changed support and render a box, arrow, typed phrase, handwriting, or doodle directly on the source image. We retain both the composited control image and its editable mark layer. Third, a logit-based VLM filter removes pairs whose target does not satisfy the requested transformation.

The retained metadata serves two distinct purposes. The target image supervises flow matching, while the audited edit box is rasterized only for the attention auxiliary loss. Raw dataset instructions are not passed to the prompt encoder: every sample uses the same trigger prompt, preventing language from bypassing the visual command. The resulting All-in-Image-45K contains 44,981 examples spanning native and high-resolution training scales. We train separate experts for in-image text, visual markers, directional arrows, and doodle-style controls; closely related box and circle text subsets share one expert.

Visual syntax and rendering. Boxes and circles specify regions, strikethroughs request removal, and arrows indicate directed movement. Short text supplies semantic details such as an attribute change, while doodles specify the intended shape or appearance. During rendering, we vary font style, size, stroke width, color, and placement to expose experts to different realizations of the same instruction. Annotations vary in opacity and color, and arrow endpoints are chosen to identify the intended object and direction unambiguously.

Semantic quality filtering. A VLM auditor compares the marked source and candidate target, checking execution of the requested edit, removal of artificial instruction marks, preservation of unrelated regions, and absence of visible artifacts. Its response begins with a Yes or No verdict. Let $P(\mathrm{Yes})$ and $P(\mathrm{No})$ denote the normalized probabilities of these verdict tokens. We compute the log-odds confidence

$$s_{\mathrm{filter}} = \log P(\mathrm{Yes}) - \log P(\mathrm{No}), \tag{12}$$

and discard pairs below a threshold selected on a manually inspected validation subset. The accepted target provides image-level supervision; the localized edit region provides the auxiliary attention target.

Editing-operation mixture. The expanded corpus retains approximately the same operation mixture used throughout data construction: 34% object or content addition, 28% object or region removal, and 38% attribute or content modification. These categories describe the primary requested operation, whereas expert families are defined by the visual instruction used to express it.

Annotation forms and image resolutions. Table 5 summarizes the approximate mixture maintained as the corpus is expanded to 44,981 examples. Annotation-form categories describe the primary visual representation of an instruction; they are distinct from the four expert families used for training. Resolution buckets refer to the nominal long side of each image.

Table 5: Approximate training-data composition. Shares are rounded; annotation forms and resolution buckets are separate views of the corpus.

Annotation form	Share	Resolution	Share
Rendered text	22%	512	22.6%
Handwriting / freehand text	16%	1K	53.1%
Bounding box	24%	1.5K	16.3%
Doodle	20%	2K	8.0%
Arrow / pointer	18%		

Composed controls. Selective execution training uses inputs containing instructions from other families, either alone or alongside the expert's own instruction. Each target executes only that expert's instruction and removes its mark, preserving all other instructions and their marks. If the expert's instruction is absent, the target is the unchanged marked input. Each expert is trained independently on these targets with flow matching. At inference, ARR receives one flattened image containing every authored mark; every selected expert reads this same raster together with the shared trigger prompt and generated-token trajectory. Expert assignment is known from the requested mark families, but no expert-specific crop, rendered layer, or edit-region mask is supplied. The evaluator retains the annotated regions only to score execution and preservation.

All-in-Image-Bench. All-in-Image-Bench contains 600 manually checked examples with no training overlap: 360 single-control cases and 240 composed cases. Each composed case contains instructions from exactly two active expert families. Composite examples emphasize spatially separated edits so that every requested behavior has an unambiguous support and partial execution can be identified. We report per-control execution, strict all-control success, edit locality, and preservation outside the union of edited regions. Region annotations are available to the evaluator but hidden from all automatic inference methods.

Manual benchmark verification. Each example is checked for an unambiguous visual instruction, an identifiable target object or region, and a well-defined edit that requires no hidden information outside the image. For composed cases, both instructions must be individually interpretable and jointly executable. Benchmark examples are excluded from both stages of expert training.

C Implementation and Evaluation Protocols

C.1 Training Configuration

Table 6 summarizes the training settings. Each expert uses the same optimizer configuration in both stages: 1,500 updates for transformation and locality learning, followed by 1,000 updates for selec-

tive execution. The pretrained encoder, VAE, and backbone remain frozen. Attention supervision is applied in the first stage; the second stage uses the expert-specific flow-matching targets in Eq. 7.

Table 6: Training and inference configuration.

Item	Setting
Main backbone	Qwen-Image-Edit-2509
Adaptation	Condition-side K/V LoRA
LoRA rank / scaling	$r = 64$, $\alpha = 128$
Optimizer	AdamW
Learning rate	3×10^{-4}
Batch size	16
Weight decay	0.01
Training resolution	512 and 1024
Timestep sampling	Uniform
Stage 1 updates	1,500 per expert
Stage 2 updates	1,000 per expert
Hardware	8 NVIDIA V100 GPUs
Attention-supervised layers	20, 40, 59
Sampled image queries	64
Initial attention-loss weight	0.02, rescaled by FM loss
Denoising steps / CFG	20 / 1

C.2 Evaluation Inputs and Scoring

Auxiliary backbone comparisons. The main results and controlled ARR ablation use Qwen-Image-Edit-2509. Supplemental tables also report LongCat condition-route results as separate interface tests; they are not pooled with the Qwen results.

Locality and execution. Attention concentration measures where generated-image queries retrieve visual-condition keys, while image-level qualification measures whether the requested edit is executed.

Benchmark allocation. The main benchmark comparison evaluates 360 single-control examples with three random seeds. The composition comparison evaluates a separate set of 240 examples, each containing two controls from known expert families. The selective-execution diagnostic uses 40 examples, with 10 examples for each of the four instruction families. Executable marks are activated by the fixed visual-edit trigger.

Continuous VLM-judge scores. For semantic instruction-following scores, the fixed Qwen3-VL-4B judge receives the marked input, edited output, and task type. It evaluates target selection, execution of the requested transformation, preservation of unrelated regions, and visual plausibility. The judge is prompted to return one digit from 0 to 9. We extract the logits ℓ_k of these ten digit tokens in a single forward pass and compute

$$p_k = \frac{\exp(\ell_k)}{\sum_{j=0}^{9} \exp(\ell_j)}, \qquad S_{\text{judge}} = \frac{10}{9} \sum_{k=0}^{9} k\, p_k. \tag{13}$$

The resulting score lies on a 0–10 scale and avoids sampling variability in the decoded verdict. Strict execution is evaluated separately using binary control-completion judgments; for two-control examples, both judgments must be positive.

C.3 Composed-Control Protocol

The composed-control evaluation follows a frozen 240-case protocol. Each image receives two binary control-completion judgments, a mark-removal judgment, and a background preservation measurement outside the authorized edit union. Strict success (All-2) is positive only when both controls succeed; partial execution does not receive full credit.

Completion and edit-support metrics. For the two requested controls, let $c_1, c_2 \in \{0, 1\}$ denote the fixed per-control judgments. Strict success is All-2 $= \mathbf{1}[c_1 = c_2 = 1]$. To measure whether the image changes where the requested edits occur, we form a spatial change support from the aligned source and output images:

$$D_{\text{edit}} = |I_{\text{out}} - I_{\text{src}}|, \qquad M_{\text{edit}} = \mathbf{1}[D_{\text{edit}} > \tau]. \tag{14}$$

Here D_{edit} is the absolute pixel difference between the aligned source and output, and thresholding yields the binary spatial edit mask M_{edit}. The threshold τ is fixed across methods and examples. For box-guided edits, the reference mask is rasterized from the annotated box coordinates. For composed edits, G is the union of the annotated edit regions. The reported Edit-IoU is

$$\text{EditIoU} = \frac{|M_{\text{edit}} \cap G|}{|M_{\text{edit}} \cup G|}. \tag{15}$$

Thus Edit-IoU measures spatial alignment of the actual image change, rather than whether an edited image merely satisfies a textual description. G is used only for evaluation and is never supplied to the model.

Preservation and cost. Let G^{+16} be G dilated by 16 pixels. We compute

$$\text{BG-PSNR} = \text{PSNR}\big(I_{\text{out}}|_{\Omega \setminus G^{+16}}, I_{\text{src}}|_{\Omega \setminus G^{+16}}\big). \tag{16}$$

Outside-region LPIPS is computed on the same retained pixels. Mark removal and artifact freedom are binary image judgments. Measured generation time is normalized by a single native edit under the matched 20-step schedule.

We stratify results into visual-mark+text, doodle+text, and force+text before macro averaging. The force slice is additionally grouped by displacement. Matched comparisons hold experts, the shared multi-mark condition, initial noise, scheduler, and seed fixed while changing the composition method: parameter merging, independent prediction mixing, or joint attention with ARR.

ARR computes the base key and value projections once, adds the selected experts' LoRA increments, and concatenates the resulting banks for one joint attention operation. The experts share the backbone computation and one latent trajectory. Independent prediction mixing instead performs two full model forwards from the same latent at every denoising step and averages their velocity predictions before updating that latent. Its $2.00\times$ cost in Table 4 denotes forward-pass count; the $1.00\times$ and $1.13\times$ entries for parameter merging and ARR report normalized generation time.

Seeded control-pair subset. Table 7 reports 20 cases per control pair at three seeds, separately from the 240-case aggregate in Table 4. In this smaller analysis, ARR ties the universal-soft baseline on doodle+text and visual+text completion and trails it slightly on force+text. Translate–edit attains higher All-2 completion on all three pairs, while ARR has the lowest outside-region LPIPS.

Table 7: Three-seed All-2 completion by control pair. Each family contains 20 cases evaluated at three seeds. Macro Outside LPIPS is reported in the last column.

Method	Doodle+text	Force+text	Visual+text	LPIPS↓
Universal soft	51.7	16.7	63.3	0.382
Translate–edit	**61.7**	**25.0**	**75.0**	0.432
ARR	51.7	15.0	63.3	**0.381**

D Attention Supervision and Response Diagnostics

Fig. 8 motivates both the routing strength and its adaptive weighting. The matched sweep in Fig. 8(a) places $\lambda_{\text{attn}} = 0.02$ near the empirical locality–preservation knee, capturing a large localization gain before outside-region changes increase substantially. A fixed coefficient, however, does not maintain a fixed optimization balance: the FM loss decreases rapidly as training progresses, whereas the routing objective decays more slowly [Fig. 8(b)]. We therefore dynamically rescale the routing weight according to the FM loss, reducing its relative pressure as the primary editing objective converges. This adjustment occurs after locality has largely formed for most control families [Fig. 8(c)]. Force responses localize more gradually than those of the other families.

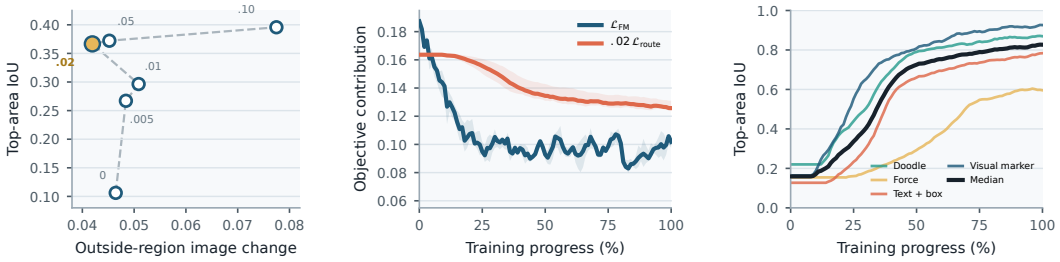

Figure 8: Routing strength and adaptive weighting. (a) A matched routing-weight sweep reveals a locality–preservation trade-off, with $\lambda_{\text{attn}} = 0.02$ lying near the empirical knee. (b) During training, the FM objective decreases substantially faster than the routing objective, so a fixed routing coefficient would exert progressively stronger relative pressure as the primary editing objective converges. (c) Response locality approaches its final range early for most control families, motivating FM-loss-conditioned reweighting after localized execution has formed.

D.1 Family-Level Responses and Adaptation Scope

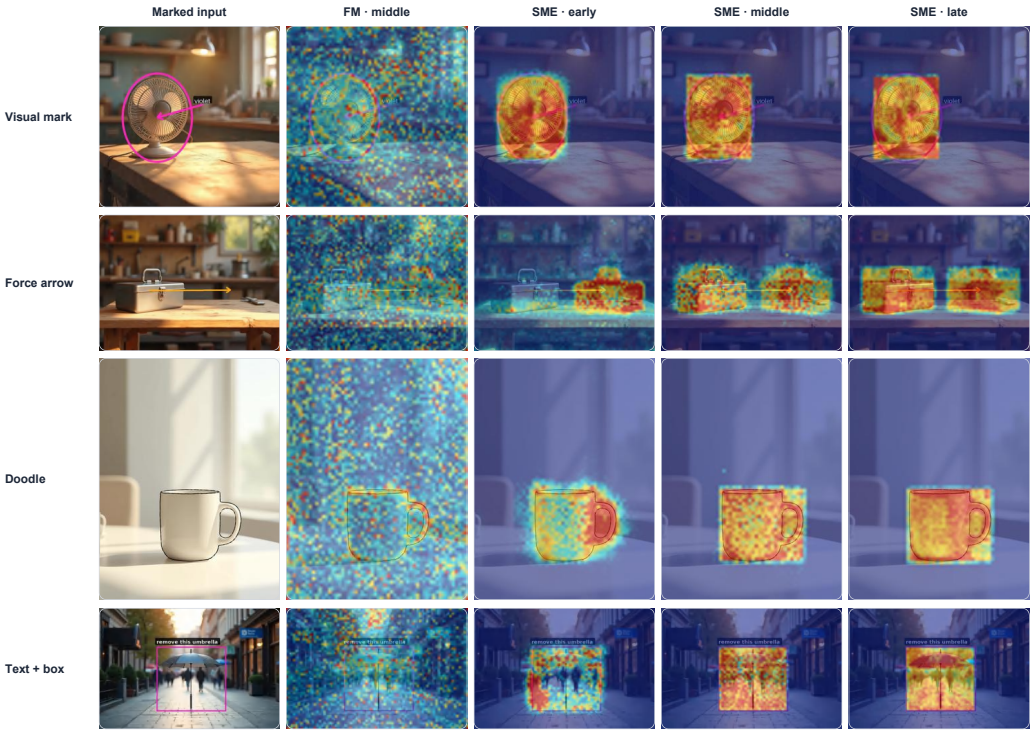

Figure 9: Routing supervision produces family-specific spatial support. Rows span visual-marker, force, doodle, and text-and-box controls; columns compare the marked input, the FM-only middle-layer response, and early, middle, and late routing-supervised responses. Instance, path, shape, and region support emerge through the same image-query-to-condition-key interaction.

Table 8(a) reports response IoU for each control family under matched FM-only and routing-supervised training. Panel (b) gives the existing adaptation-scope comparison: K/V adaptation with routing reaches 73.0% average edit execution, compared with 72.3% for Q/K/V/O adaptation with routing on the same held-out tasks.

Table 8: Routing supervision and adaptation scope. (a) Matched K/V continuations use the same initializer, data order, and number of FM updates; only routing supervision changes. Response IoU is measured at timestep 500 on 256 held-out examples per family. (b) Edit execution by adaptation scope on 64 held-out examples per task, evaluated by the fixed target-blind judge.

(a) Locality by control family

Family	IoU_{FM}	$\text{IoU}_{+\text{routing}}$
Box text	.104	**.783**
Circle text	.111	**.786**
Doodle	.168	**.888**
Force	.140	**.612**
Visual marker	.156	**.924**

(b) Execution by trainable route (%)

Route	Marker	Force	Doodle	Text	Avg.
FM expert	53.1	6.3	**100.0**	**93.8**	63.3
Q/K/V/O + routing	**76.6**	26.6	**100.0**	85.9	72.3
K/V + routing	70.3	**29.7**	98.4	**93.8**	**73.0**

Compound spatial responsibilities. Fig. 10 shows a three-mark scene. The hat, mug relocation, and pancake experts produce different spatial preferences from the *same* compound input. Joint attention over their K/V banks allocates attention mass across experts at each generated location, yielding all three edits on one trajectory. The maps visualize expert responses, not separate masks supplied at inference.

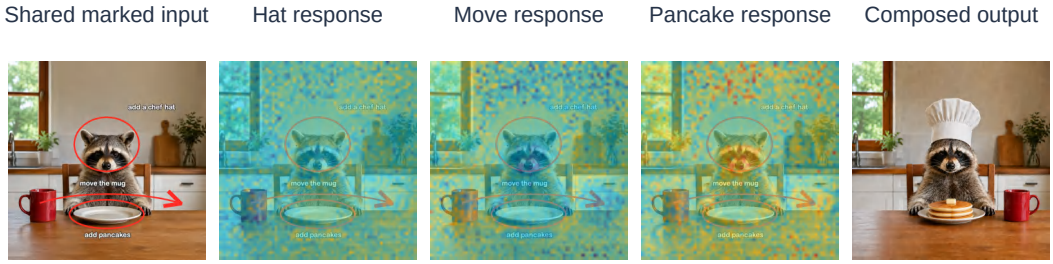

Figure 10: Expert-specific responses under a compound instruction. One image carries three requests; the three response maps highlight distinct supports as their K/V banks participate in joint attention. No expert-specific crop or target mask is given at inference.

E Composition Analysis

E.1 Per-Expert Capability Retention

We evaluate the same expert checkpoints independently and under compound inference. Table 9 measures per-family editing accuracy and the ratio retained under composition. Average per-family retention is 84.7%, with text/box, visual-marker, and doodle experts retaining about 89–91% of their individual accuracy. These per-control measurements complement the strict all-control success reported in Table 4.

Table 9: Behavioral capability retention. Accuracy and retention are percentages. The last row averages each column across families.

Expert family	Single↑	Composed↑	Retained↑
Text / box	93.8	85.0	90.6
Visual marker	70.3	62.0	88.2
Force / arrow	29.7	21.0	70.7
Doodle	98.4	88.0	89.4
Macro avg.	73.1	64.0	84.7

E.2 Overlapping Edit Regions and Early Layout Initialization

When spatial responses overlap. Force instructions can affect a broad image region as an object moves between locations. Their spatial responses may therefore overlap those of another expert,

making the division of editing responsibilities less clear at the start of generation. This case differs from composing edits at well-separated locations: the experts must establish a compatible spatial arrangement as well as carry out their individual transformations.

Establishing the initial layout. For overlapping edits, we use the first two denoising steps to establish an initial layout. At each of these steps, every selected expert independently predicts a velocity field from the same current latent, with only its own LoRA loaded. We combine the predictions regionally according to the expert responses, selecting the stronger velocity prediction within each region, and use the resulting field to update the shared latent. From the third step onward, generation proceeds with the joint key–value attention in Eq. 9.

F Trigger Design and Text-Guided Editing Compatibility

A task switch that keeps the edit in the image. We use the same short, nonempty trigger for visual-editing examples. It selects visual editing without specifying the object, operation, or location; those remain encoded by the authored marks. Ordinary text-guided editing instead receives the user's natural-language instruction, without the visual-edit trigger. This separates the new interface from the editor's existing text interface while retaining both in the same adapted model.

Why retain a nonempty trigger? The empty text condition has a separate role in classifier-free guidance (CFG): it provides the reference prediction against which a text-conditioned prediction is contrasted Ho & Salimans (2022). Writing $v(c)$ for the prediction at a fixed noisy state and image condition, CFG takes the form

$$v_{\mathrm{CFG}} = v(\varnothing) + g\big[v(c) - v(\varnothing)\big], \tag{17}$$

where c is the text condition and g is the guidance scale. Using empty text as the visual-edit command would train this reference condition to execute marks, changing its role in the guidance difference. A dedicated trigger gives visual editing its own text condition instead. Our visual-editing evaluation uses $g = 1$; this design also keeps the interface compatible with the editor's ordinary guided use.

Retention of text-guided editing. Table 10 evaluates ordinary text-guided editing on 20 MagicBrush sequences with natural-language instructions and no visual marks or visual-edit trigger. Each trained Qwen expert is evaluated separately on the same sequences, and the adapted-model row reports the mean across experts. Final-turn evaluation scores the last output of each sequence; all-turn evaluation includes intermediate outputs. The adapted model remains close to the frozen editor on reconstruction and image-similarity metrics, while CLIP-T improves from .3283 to .3297 in final-turn evaluation and from .3162 to .3224 across all turns. These results show that learning to execute visual instructions retains the model's ordinary text-guided editing ability.

Table 10: Compatibility with ordinary text-guided editing. Each Qwen expert is evaluated independently on the same 20 MagicBrush sequences; adapted scores are averaged across experts. No visual mark or visual-edit trigger is supplied. The final-turn protocol evaluates the last edit of each sequence, while all-turn evaluates every intermediate turn.

Setting	Editor	N	$\ell_1 \downarrow$	$\ell_2 \downarrow$	CLIP-I↑	DINO↑	CLIP-T↑
Final-turn	Frozen Qwen editor	20	.1942	.0899	.8406	.7385	.3283
Final-turn	Qwen experts (mean)	20	**.1941**	.0914	.8345	**.7668**	**.3297**
All-turn	Frozen Qwen editor	20	**.0947**	**.0403**	.9232	.8809	.3162
All-turn	Qwen experts (mean)	20	.0953	.0411	**.9253**	**.8854**	**.3224**

G Additional Qualitative Results

Figures 11–16 expand the qualitative evaluation from individual controls to every pair of families, repeated instructions within one family, and three simultaneous controls. Each example places the marked input beside the corresponding output.

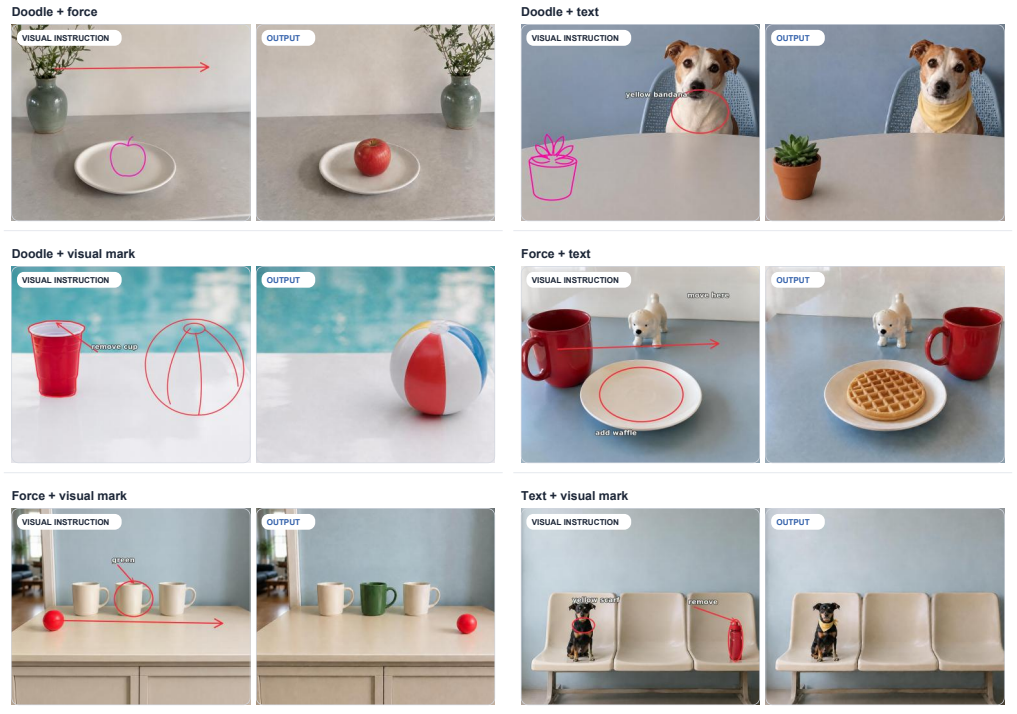

Figure 11: One composition rule covers every pair of control families. The matrix enumerates all six unordered pairings among doodle, force, text-and-box, and visual-marker controls. Each multi-marked input is paired with its corresponding edited image.

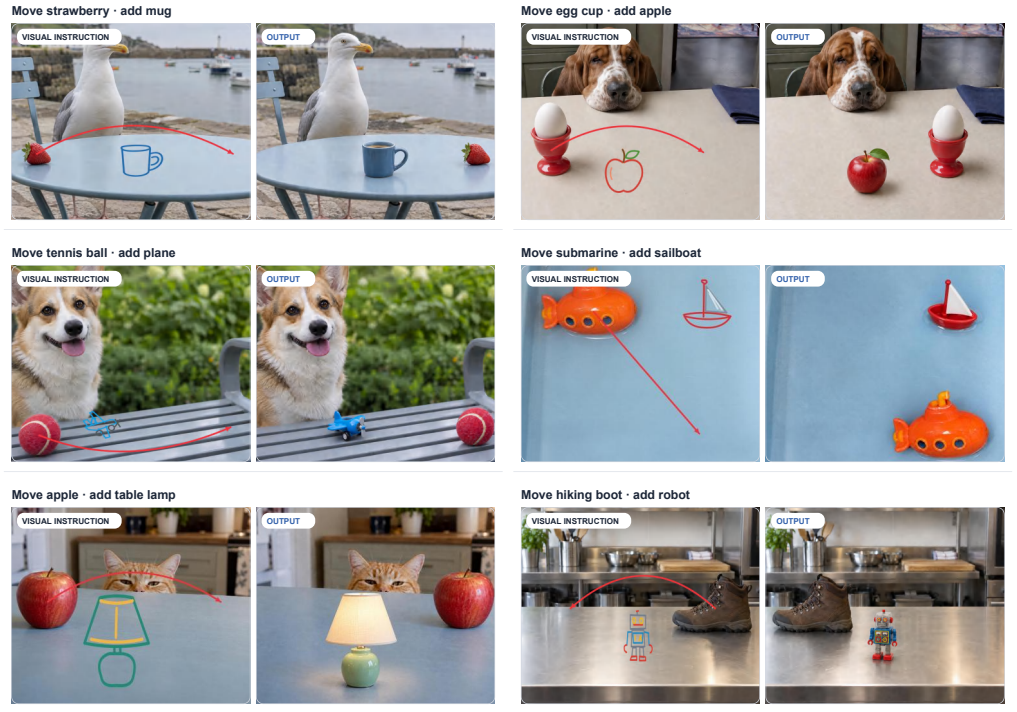

Figure 12: Force and doodle controls compose across varied scenes. The first gallery pairs six jointly marked inputs with their corresponding edited images across object, animal, and landscape scenes.

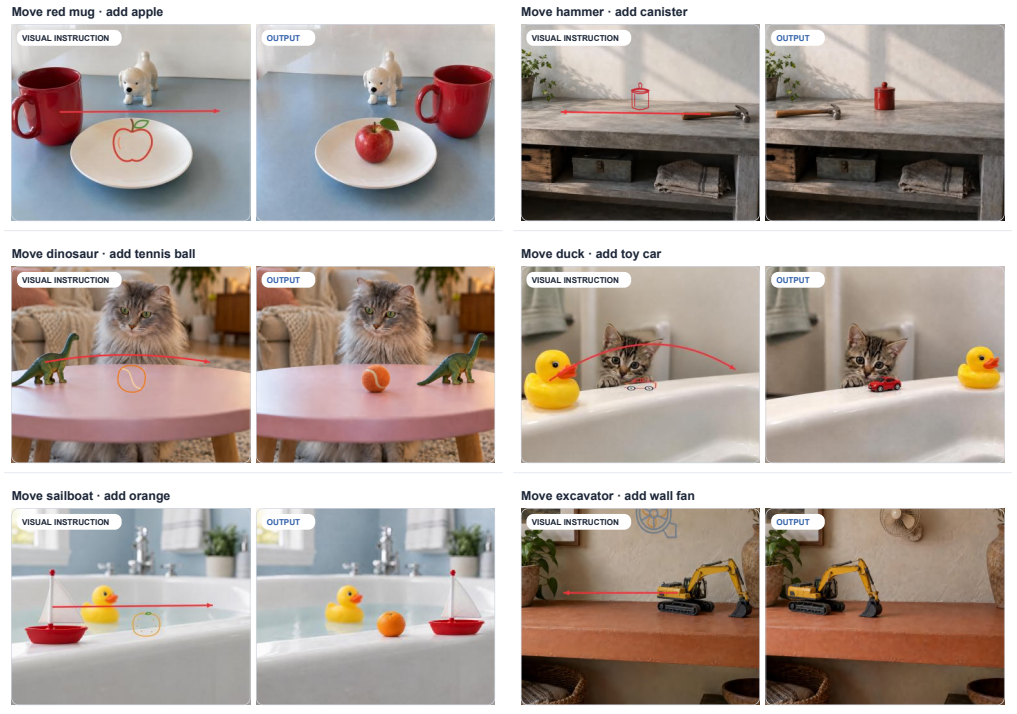

Figure 13: The same force–doodle composition extends across edit types. Six additional examples combine directed motion with doodle-conditioned appearance or insertion while preserving the shared scene.

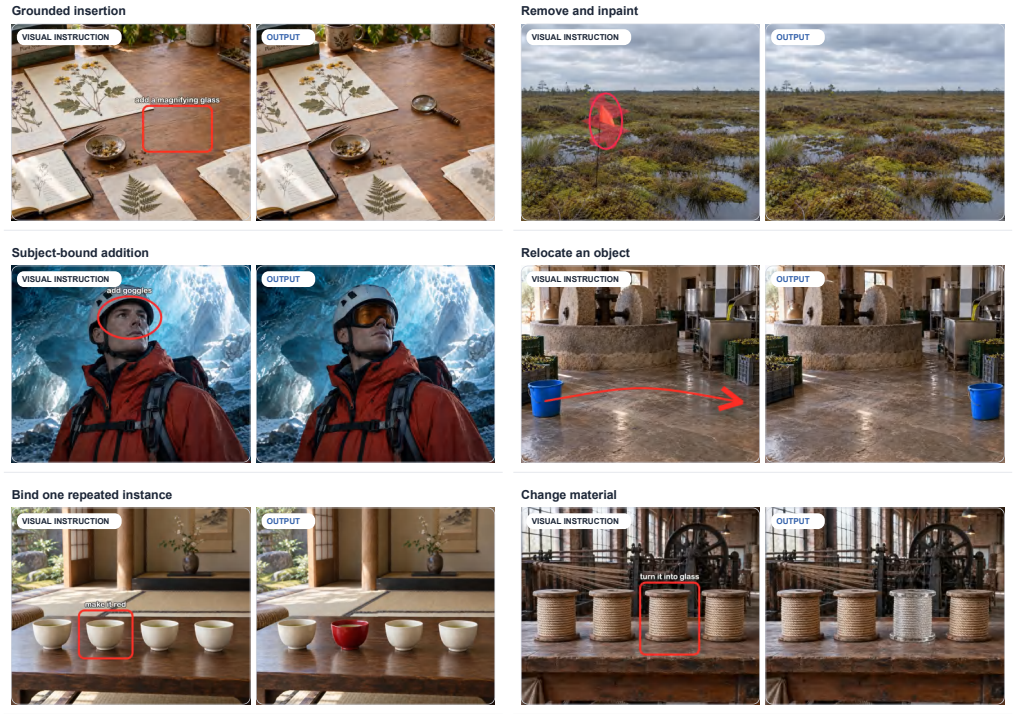

Figure 14: Sparse experts execute diverse individual visual relations. The gallery covers recoloring, removal, insertion, relocation, resizing, and doodle-conditioned rendering, with the marked input and output shown side by side.

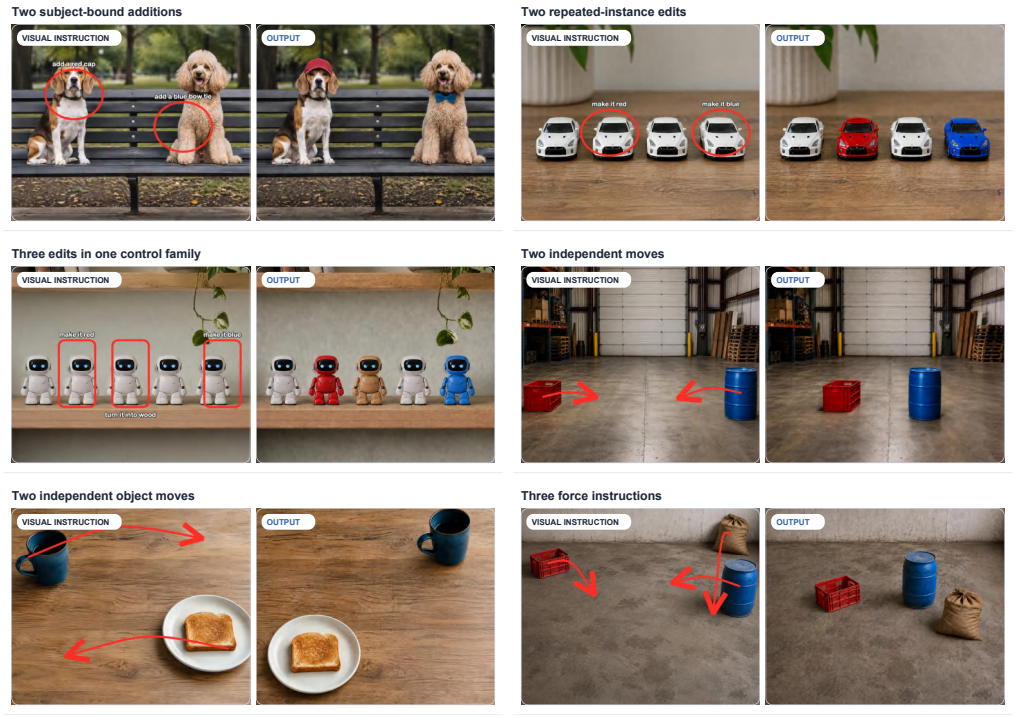

Figure 15: One expert can execute several instructions from its own family. Each example contains multiple spatially separated requests that share a control family; the paired edited image shows the requested changes in their respective regions.

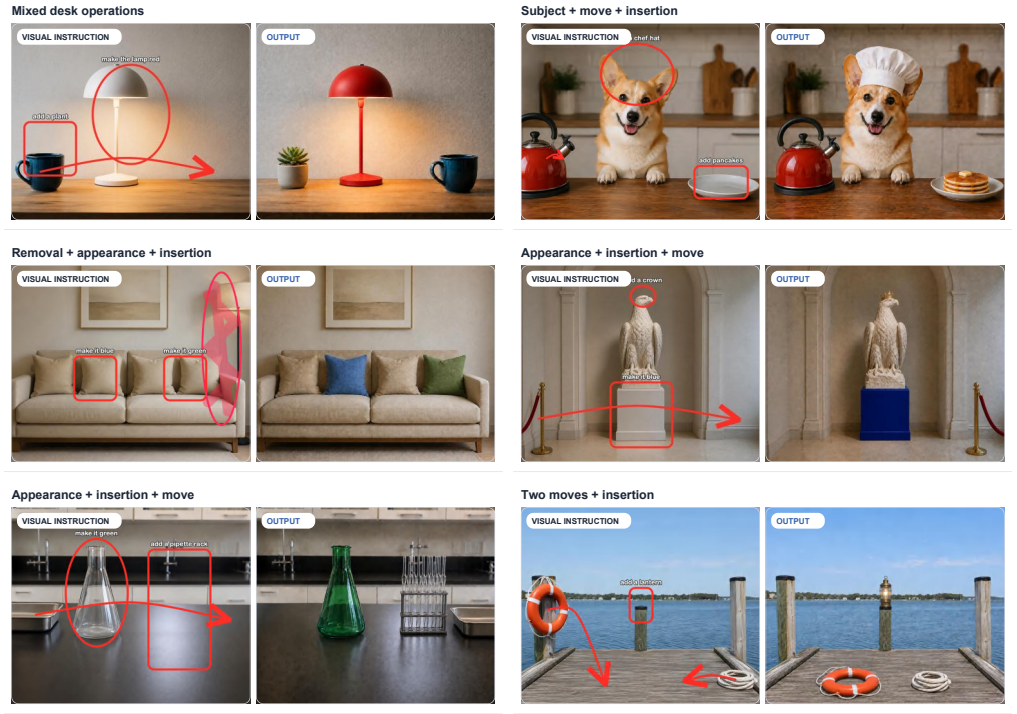

Figure 16: Response-space allocation extends to three simultaneous controls. Each marked input combines three requested relations, and the corresponding edited image shows their joint target within one shared scene.

H Interface Robustness and Spatial Correspondence

H.1 Human-Drawn Instructions and Spatial Disambiguation

Human-drawn instructions. The paired 300-task study holds source, intent, and target fixed while changing only the annotation style. Ten volunteers used a mouse or stylus without restrictions on handwriting, stroke width, arrow shape, or box geometry. The condition routes are evaluated zero-shot: none of these human annotations is used for training. Table 11 shows that the LongCat route retains a .83 human-drawn CLIP-GT average versus .88 on the regular counterparts, while the native editor falls to .58. The remaining errors concentrate in illegible text, unclear arrow endpoints, and overly broad boxes.

Table 11: Zero-shot robustness to human-drawn instructions. The 300-task paired set holds the source, intent, and target fixed between regular and human-drawn annotations. Ten volunteers authored 100 handwritten instructions, 100 arrow–text controls, and 100 irregular box–text controls. CLIP-GT is reported on $[0, 1]$; MRR is measured on the human-drawn inputs.

Editor	Regular avg.	Handwriting	Arrow+text	Box+text	HD avg.	HD MRR (%)
Qwen native	.71	.61	.55	.58	.58	71
Condition route (Qwen)	.86	.82	.79	.81	.81	93
LongCat native	.70	.61	.54	.58	.58	72
Condition route (LongCat)	**.88**	**.83**	**.82**	**.83**	**.83**	**94**

Separated text and repeated objects. Table 12(a) asks whether a mark contributes spatial information when the intent is already available through the text channel. Adding a marker to the native editor raises target accuracy from 72.4% to 79.6% and reduces wrong-object edits from 18.9% to 10.8%. Under the same marker-plus-text input, the condition route reaches 90.5% target accuracy; placing the short request in the image raises it to 91.2%. Panel (b) makes the ambiguity explicit with 2–10 similar objects, where the LongCat route reaches 92.4% target accuracy and 94.5% non-target preservation.

Table 12: Spatial correspondence beyond text alone. (a) The intent is supplied separately while the image contains either no mark or a spatial mark; the final row instead uses the complete in-image instruction. (b) Each scene contains 2–10 similar objects and only the marked instance should change. MRR is marker-removal rate.

(a) Separated-text diagnostic

Setting	CLIP-GT	Target↑	Wrong↓	MRR↑
Native + text only	78.2	72.4	18.9	–
Native + mark + text	83.4	79.6	10.8	89.1
VLM pipeline + mark + text	90.4	89.8	3.6	97.1
Condition route + mark + text	90.9	90.5	3.0	98.2
Condition route + in-image text	**91.3**	**91.2**	**2.7**	**98.4**

(b) Hard multi-object scenes

Editor	Target↑	Wrong↓	Pres.↑	MRR↑
Qwen native	63.2	24.7	71.9	89.4
LongCat native	64.5	23.8	73.1	90.1
OmniGen2	66.7	21.2	75.6	90.3
VLM pipeline	88.9	4.5	91.3	97.0
Condition route (Qwen)	91.2	3.1	93.6	98.0
Condition route (LongCat)	**92.4**	**2.6**	**94.5**	**98.5**

Target, Wrong, Pres., and MRR entries are percentages.

H.2 Generalization to Unseen Visual-Text Instructions

Figure 17 presents qualitative examples of editing with unseen visual-text instructions, including plain rendered text, handwriting, and stylized text. The lower comparison contrasts the frozen editor with the adapted route on the same instructions.

H.3 Natural Scene-Text Preservation and Authored-Mark Removal

On 300 scene-text lookalike examples, the LongCat condition route preserves 291 of 300 natural scene-text instances (97.0%) while removing 294 of 300 authored instruction overlays (98.0%). These results show that the route can distinguish authored instructions from visually similar text already present in the scene.

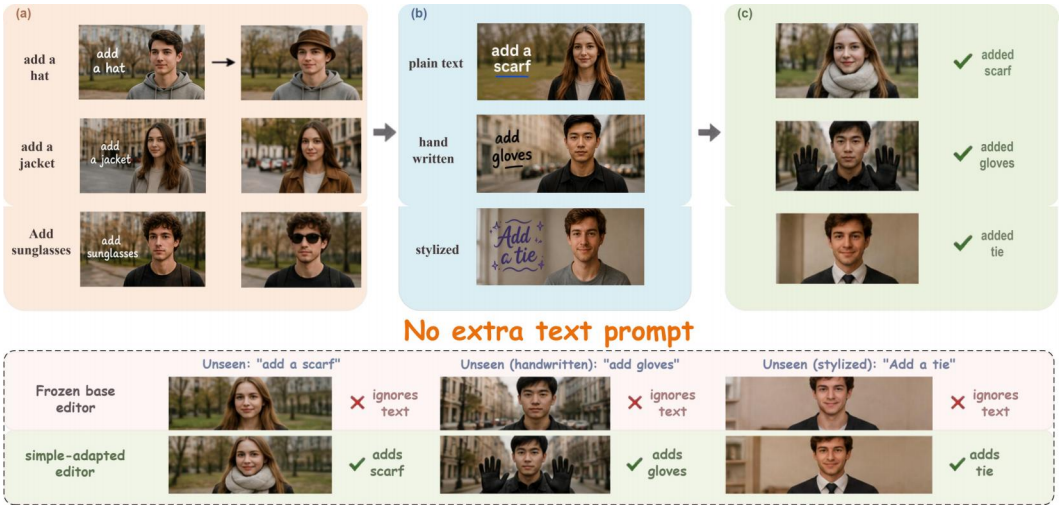

Figure 17: Qualitative generalization to unseen visual-text instructions. Panel (a) shows rendered training pairs; panels (b–c) show unseen plain, handwritten, and stylized instructions and their corresponding outputs, without an edit-specific text prompt. In the lower examples, the frozen editor leaves the requested edits unexecuted, while the adapted route performs them.

I Evaluation on VP-Bench

Benchmark and metrics. We evaluate **VEGA** on the full VP-Bench Yi et al. (2026) benchmark. The evaluation measures instruction fidelity (IF), content consistency (CC), visual realism (VR), and spatial precision (SP), each on a five-point scale. Table 13 compares **VEGA** with Qwen-Image-Edit-2509, Nano-banana, and FlowInOne. All methods are evaluated by three VLMs: Gemini 3, GPT-5.2, and Qwen3.5. We report the mean score across the three evaluators for each dimension. The final column averages the four dimensions.

Results. **VEGA** achieves the highest content consistency (4.29) and spatial precision (3.51) among the compared methods, exceeding the strongest baseline in these dimensions by 1.33 and 0.19 points, respectively. Its visual realism score of 4.17 ranks second to Nano-banana (4.31). The overall average reaches 3.77, compared with 3.39 for Nano-banana and 3.14 for FlowInOne. These results highlight **VEGA**'s ability to localize edits while preserving image content; instruction fidelity remains below Nano-banana and FlowInOne.

Table 13: Evaluation on VP-Bench. IF: instruction fidelity; CC: content consistency; VR: visual realism; SP: spatial precision. Each dimension is scored out of five; Avg. is their mean. Scores for all methods are averaged across three VLM evaluators: Gemini 3, GPT-5.2, and Qwen3.5. Best and second-best results are marked in **bold** and underline, respectively.

Method	IF↑	CC↑	VR↑	SP↑	Avg.↑
Qwen-Image-Edit-2509	2.68	2.29	3.36	1.74	2.52
Nano-banana	**3.34**	<u>2.96</u>	**4.31**	2.93	<u>3.39</u>
FlowInOne	<u>3.28</u>	2.87	3.09	<u>3.32</u>	3.14
VEGA	3.09	**4.29**	<u>4.17</u>	**3.51**	**3.77**

J Human Preference Study

Study design. We conduct a human preference study on 100 sets of editing results. Each set contains seven candidate outputs, one from each of InstructPix2Pix, FLUX, LongCat-Image-Edit, Qwen-Image-Edit, OmniGen2, Nano-banana, and **VEGA**. Participants select their preferred output from each set. We collect one selection per set and report each method's share of the 100 selections in the final column of Table 1.

Results. **VEGA** receives 22% of the selections, ranking second among the seven editors, three percentage points behind Nano-banana (25%). LongCat-Image-Edit receives 16%, Qwen-Image-Edit and OmniGen2 each receive 14%, and FLUX receives 9%; InstructPix2Pix receives no selections. The preference ranking complements the automatic evaluation: **VEGA** leads in strict execution and edit localization, while its outputs also receive a larger preference share than the other five comparison editors.