SMART-Editor: A Multi-Agent Framework for Human-Like Design Editing with Structural Integrity

Ishani Mondal,[1] **Meera Bharadwaj,**[1] **Ayush Roy,**[1]
Aparna Garimella,[2] **Jordan Boyd-Graber**[1]
[1] University of Maryland, College Park, [2] Adobe Research Lab
{imondal@umd.edu, garimella@adobe.com, jbg@umiacs.umd.edu}

Abstract

Despite significant progress in natural image editing with state-of-the-art MLLMs, compositional layout and content editing for structured visual domains (e.g., posters, websites) remains under-explored. We introduce **SMART-Editor**, a multi-agent framework for compositional editing for structured images like posters or websites. Unlike prior models that focus on isolated local edits, **SMART-Editor** maintains global coherence through two complementary strategies: Reward-Refine, an inference-time reward-guided refinement method, and RewardDPO, a training-time preference optimization approach leveraging reward-aligned layout pairs. To evaluate performance, we introduce **SMARTEdit-Bench**, a benchmark of cascading multi-step edit instructions that are implicit in nature yet require layout and semantic-consistency preserving reasoning about edit order to preserve spatial and semantic consistency. Both automatic and human evaluations confirm that reward-guided planning produces semantically consistent and visually coherent edits, beyond what single-shot VLMs can generate.

1 Introduction

Compositional structure and content editing represents a fundamental challenge in multimodal reasoning, as it requires coordinating visual layout with semantic content across diverse tasks, including poster restructuring, webpage reflow, and document editing (Suri et al., 2024; Zhu et al., 2024; Gani et al., 2024; Tanaka et al., 2024; Lin et al., 2024). These edits often go beyond simple object manipulations and require coordinated adjustments across spatial, semantic, and stylistic dimensions (Jia et al., 2024; Feng et al., 2023; Lin et al., 2023). Instructions like "Insert a video section at the middle" (Figure 1) require cascading edits such as identifying an insertion point in layout hierarchy, shifting subsequent sections to preserve flow, aligning with adjacent regions, and maintaining coherence of semantic information throughout the poster.

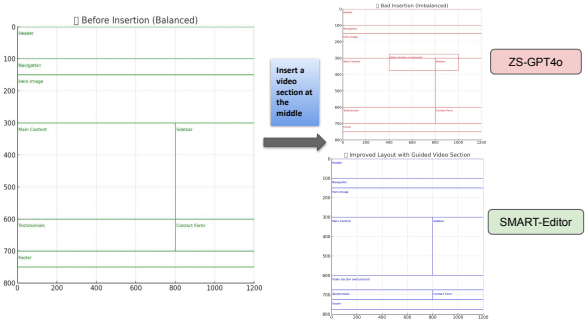

Figure 1: Unlike base LLMs, ***SMART-Editor*** *performs structured layout edits with compositional reasoning, preserving global alignment and semantic consistency (Clear Figures in Figure 11 and Figure 12).*

When *humans* learn how to draw or typeset, they learn about the basics of composition and balance (Golomb, 1987), but it is still not clear whether large language models (LLMs) or visual language models (VLMs) can make edits that keeps the layout organized and the meaning consistent. These tasks are critical for scientific communication (Lozano, 2014), where even moving a single section to the wrong place can confuse the reader and make the poster/website much harder to digest.

To explore this challenge, we introduce **SMART-EDIT**—an edit that (a) adheres semantically to the instruction, (b) minimizes disruption to unrelated content, (c) preserves spatial coherence and alignment, and (d) anticipates and resolves cascading effects (formalized in Section 2). Recent interactive editing frameworks such as DragGAN (Pan et al., 2023) and DragDiffusion (Shi et al., 2024) enable spatially precise manipulations through point-based dragging, while GenArtist (Wang et al., 2024) extends this idea to cascaded natural-image edits via agentic decomposition and planning trees.

However, these approaches remain fundamentally object-centric: they can adjust a nose, reshape a car body, or recolor a patch, but they ignore layout-wide dependencies. In structured images such as posters or webpages, adding a section like

Findings of the Association for Computational Linguistics: EACL 2026, pages 3219–3245

Results ripples through the whole design—shifting text blocks, rebalancing whitespace, and preserving column order. Unlike natural images, where edits are judged at the pixel level locally (Brooks et al., 2023), structured layouts require global reasoning about hierarchy and flow. Therefore, we introduce a new benchmark, **SMARTEdit-Bench** (Section 3), that evaluates compositional edit reasoning in LLMs and VLMs across scientific posters and websites. Each instance in the benchmark features a real-world or curated edit sourced from datasets such as SciPostLayout (Tanaka et al., 2024), Design2Code (Si et al., 2025) comprising an initial scene, a human-verified SMART-EDIT instruction, and a target image requiring multi-step semantic and structural updates. Unlike prior datasets that target explicitly specified edits on natural images (Table 8), we introduce **SMARTEdit-Bench**, the first benchmark designed to evaluate whether models can make globally coherent, multi-step edits that preserve *human-like design integrity* in structured domains.

Base LLMs/VLMs like LLaMA, Qwen, LLaVA, GPT4-o and Gemini-pro can follow instructions superficially where simple instruction following is required (edit adherence >= 4.0, definition in Section 2), but fail to preserve deeper layout reasoning: narrative coherence drops by over 15% during section reordering, and layout alignment degrades sharply during insert/delete operations. Cross-sectional consistency is especially fragile when instructions induce global changes (e.g., regrouping content or propagating section deletions). To address these limitations, we introduce **SMART-Editor**, a modular agentic framework that coordinates planning, execution, critique, and optimization (Section 4 and Figure 2). The **Actor** translates edit instructions into layout operations, which are applied by the **Executor**. A **Critique** then evaluates the output across structured reward axes (e.g., overlap, whitespace, coherence) and produces actionable feedback in natural language. This feedback/new action plan guides the **Optimizer**, which operates in two modes: (1) **Reward-Refine**, which uses critiques to revise the initial plan in an iterative loop until all quality checks are satisfied; and (2) **RewardDPO**, which fine-tunes the model using preference pairs filtered by reward alignment to generalize better layout reasoning policies. Across posters and webpages, SMART-Editor improves layout alignment, narrative and cross-sectional consistency compared to base LLMs/VLMs and existing editing pipelines (Figure 4, 6).

2 What is a SMART-Edit?

Inspired by the evaluation metrics used by prior layout generation works (Tanaka et al., 2024; Lin et al., 2023; Hui et al., 2024; Zhang et al., 2024), we formalize **SMART-Edit** as an edit operation that, when applied to structured or unstructured visual content, satisfies the following post-edit quality constraints in structured domain (Definitions in Appendix 12.1):

1) Edit Adherence : It verifies whether the edit instruction is faithfully executed.

2) Narrative Coherence: It ensures logical section ordering (e.g., *Methods* before *Results*) is preserved. Violations are detected by prompting GPT-4o on expected order pairs.

3) Cross-Sectional Consistency: It evaluates whether semantically linked elements (e.g., captions and figures, objects and context) remain aligned. Contradictions are flagged via entailment-based GPT-4o prompts.

4) Visual-Spatial Layout: It captures design quality through three components: overlap penalty (discouraging collisions), whitespace penalty (encouraging compactness), and alignment reward (favoring grid-aligned layouts).

3 SMARTEdit-Bench

Current benchmarks (Table 8) for layout editing inadequately model the implicit and compositional nature of real-world edits, where high-level instructions often induce coordinated changes across multiple regions, and hence we introduce **SMARTEdit-Bench**, a multi-domain benchmark covering structured inputs like posters and webpages (Table 9). Each example includes a natural language instruction, a human-edited output, and metadata annotated with edit type (A manual analysis of 200 real-world edit examples identifies five types of structural edits: Reordering, Insertion, Object Replacement, Content Grouping, and Deletion (Table 10)—which are implicit in nature (without specifying how to edit) and require cascading, multi-region updates).[1]

SMARTEdit-Bench–Webpages. We divide the dataset into two parts: (a) Real-World Edits: We ex-

[1]All source datasets are publicly available under CC-BY and MIT License.

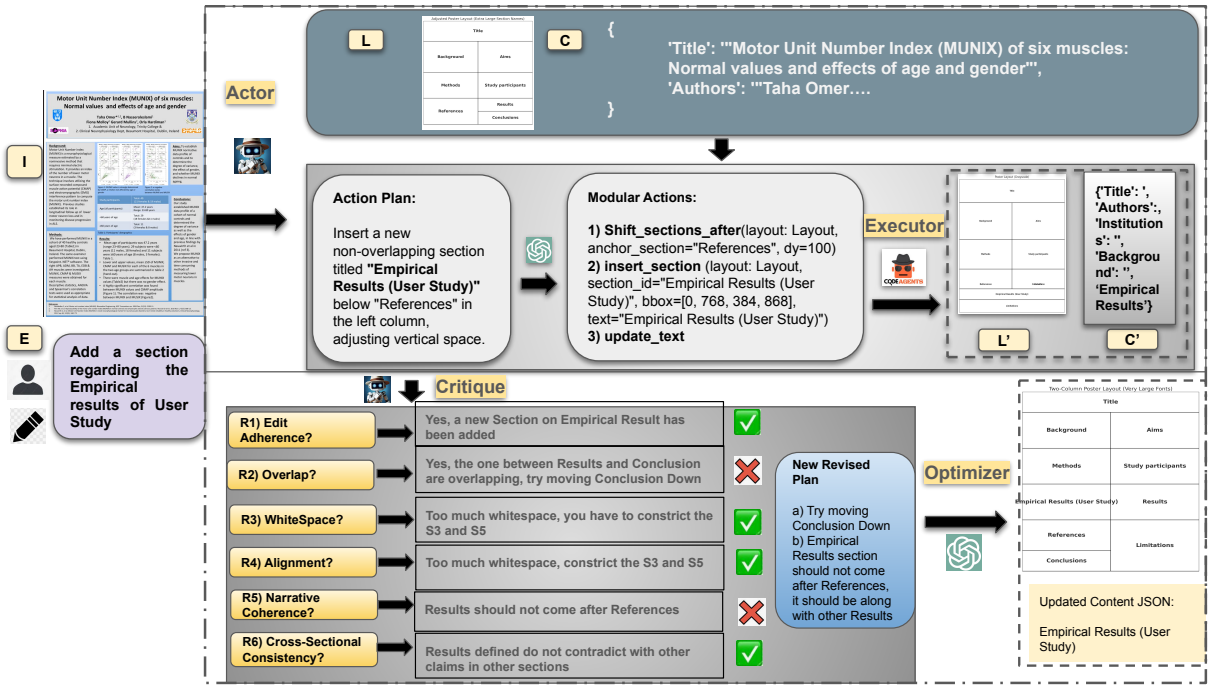

Figure 2: Overview of **SMART-Editor** (Reward-Refine) on structured image (Poster). Given a poster image, and a high-level edit instruction, SMART-Editor generates an action plan and applies geometric edits (**Actor**). The first-draft output is evaluated on several aspects (**Critique**). Any violations trigger natural language feedback used to revise layout and content in an iterative loop, enabling reward-guided editing (**Optimizer**).

tract layout-altering commits from github.io repositories by comparing HTML snapshots before and after structural changes. Stylistic edits are removed using two signals: low DOM tree edit distance and high semantic similarity, indicating minimal structural change. GPT-4V generates natural language instructions by comparing before-after screenshots. This results in 564 triplets (Before Commit, Edit Instruction, After Commit), each manually reviewed and refined by the first author to ensure they are *Smart-Edits*. (b) Synthetic Edits: Layouts are sampled from the Design2Code dataset (Si et al., 2025). GPT-4o generates implicit, high-level edit instructions. An expert annotator ensures the instruction leads to compositional layout updates; otherwise, both instruction and layout are revised manually. This yields 510 high-quality synthetic triplets.

SMARTEdit-Bench–Posters. We generate 1,200 synthetic poster edits using SciPostLayout (Tanaka et al., 2024), prompting GPT-4V/4o to propose compositional layout transformations. All outputs are manually reviewed to remove trivial or overly explicit edits. Two expert annotators label 250 human-edited posters with edit types and reasoning dimensions, forming a gold test set. An additional 2,000 synthetic edits support training.

4 SMART-Editor

To address structure-aware, instruction-guided layout editing (by adjusting positions, sizes, alignments, ordering, and grouping relationships of objects), we propose **SMART-Editor**, a multi-agent pipeline that orchestrates a trilateral collaboration among three core agents—**Actor**, **Critique**, and **Optimizer** and designed to modify spatial configurations in images while preserving semantic and visual coherence. The system takes an image I and edit instruction E as inputs, from which it generates an edited image I' following edit instruction.

1. Actor. First, the **Actor** takes a raw input image I and edit instruction E, and outputs a first-pass edited layout-content pair (L', C'). It first parses I into an initial layout $L = \{(o_i, \mathbf{b}_i)\}_{i=1}^{N}$, where o_i denotes a visual region (e.g., section type or object label), $\mathbf{b}_i = (x_i, y_i, x_i', y_i') \in \mathbb{R}^4$ is its bounding box. In parallel, it extracts associated textual content C to these regions (Details in Appendix 11.1).

To enable compositional and controllable layout editing from natural language, **SMART-Editor** first constructs a symbolic action plan based on the input layout-content pair (L, C) and instruction E. This plan is generated using a large language model and represents a sequence of structured layout op-

erations tailored to the instruction. Formally, we define the action plan as $A = [a_1, a_2, \ldots, a_k] = \texttt{LLM\_Plan}(E, L, C)$, where each action a_j corresponds to an operation drawn from a predefined *Action Registry* (e.g., `reorder`, `translate`, `group`, `insert`), as detailed in Table 11 (Refer to Actions in Figure 2). The resulting symbolic plan A is then executed by the *Executor Agent*, which applies each action in sequence to generate a revised layout-content pair (L', C'), representing the first-pass edited output. This intermediate result is then passed to the subsequent refinement stages.

2. Critique. Critique is needed because zero-shot edits often fulfil instructions literally but mangle the layout integrity. In Table 4, base GPT-4o inserts a video section without adjusting surrounding elements, causing overlaps and uneven whitespace, or deletes the Research Questions section but leaves disruptive blank space. The Critique Agent addresses this by evaluating each edited layout–content pair (L', C'), generating both a scalar reward score $R(L', C')$ across all the fine-grained dimesions and generate structured, actionable feedback for refinement (Prompts in Appendix 11.4). For *structured domains*, we use a combination of heuristic checks and LLM-based prompting (Appendix 12.1) to compute reward components (R1–R6 in Figure 2), enabling diagnosis of specific issues and suggestions for improvement (e.g., the Overlap Critic might note, "*Yes, the one between Results and Conclusion are overlapping, move Conclusion down*").

3. Optimizer. We compute the composite reward as a weighted sum of interpretable components: $R(L', C') = \sum_{k=1}^{K} \lambda_k \cdot r_k(L')$, where each r_k captures a specific quality axis and $\lambda_k \in \mathbb{R}$ is a tunable weight. If $R(L', C')$ falls below a threshold, the **Optimizer** agent explores two complementary strategies for optimizing generation under reward-defined constraints: an inference-time refinement loop-based method (Reward-Refine) and a training-time preference distillation approach (RewardDPO).

3.A) Iterative Refinement (Reward-Refine). This method enables self-corrections through iterative reward-guided inference (Algorithm 1). At iteration $t = 1$, we use beam search to sample k candidate layout-content pairs $(L_i^{(t)}, C_i^{(t)})$ conditioned on the previous state $(L^{(t-1)}, C^{(t-1)})$ and the instruction E. Based on the critique feedback, we convert the suggested improvement into a revised plan expressed in the structured *action space* defined by the *Action Registry* (Section 11). These symbolic actions guide the generation of new candidate layouts, and beam search is again employed to sample alternatives and select the best one by computing the reward for each. The best candidate under the reward function is selected as $(L^{(t)}, C^{(t)}) = \arg\max_{(L,C)} R(L)$. This process continues iteratively until all constraints are satisfied or a maximum number of refinement steps is reached. The final output is the converged layout-content pair $(L^*, C^*) = (L^{(T)}, C^{(T)})$, emulating a human-in-the-loop editor through critique-to-action planning.

3.B) Reward-Aligned Preference Optimization (RewardDPO) To address the limitations of inference-time refinement—particularly its runtime cost and inability to generalize beyond a single revision—we introduce **RewardDPO**, a training-time framework that distills reward-aligned preferences into the model through contrastive supervision over layout perturbations. The goal is to produce models that can generate high-quality edits in a single forward pass at test time, without relying on iterative reward-guided inference. **RewardDPO** takes the initial output L_0 after preprocessing in the **Actor Agent** as input, uses **Critique Agents** to generate preference data, and finally generates optimized layout L^* as output (We skip optimizing the content using DPO, as it improves well in 1 single iteration in **Reward-Refine**). It follows three steps as described below:

Supervised Finetuning. We begin with a supervised finetuning (SFT) stage, where the model f_θ is trained on triplets $(L_0^{(i)}, E^{(i)}, L_{\text{target}}^{(i)}) \in \mathcal{D}_{\text{SFT}}$ produced from Critique feedback as the supervision signal. The model is optimized using a log-likelihood objective: $\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{i=1}^{N} \log P_\theta(L_{\text{target}}^{(i)} \mid E^{(i)}, L_0^{(i)})$

Preference Pair Generation via Reward-Aligned Contrastive Supervision. Supervised fine-tuning typically exposes models to a single "ideal" layout per instruction, limiting their ability to reason about the relative quality of edits. Hence, we construct contrastive preference pairs (L^+, L^-), where L^+ is verifiably better than L^- based on a composite reward function: $R(L) = \sum_{k=1}^{K} \lambda_k \cdot r_k(L)$, which aggregates multiple quality dimensions such as layout

alignment, narrative flow, and visual plausibility.

For positive examples L^+, we primarily use gold-standard human-edited layouts when available (e.g., in SMARTEditBench). In cases where human edits are unavailable, we sample multiple candidate layouts from a base model and select the one with the highest reward score, while filtering out invalid candidates (e.g., layouts with overlapping elements, missing sections, or visual artifacts).

Negative examples L^- are created by degrading L^+ with transformation functions $T_k \in \mathcal{T}$, which are designed to introduce specific types of errors. In structured domains, these include misplacing sections (e.g., placing "Results" above "Methods"), inducing overlaps, misaligning bounding boxes, or distorting whitespace. We retain a pair only if the degradation results in a reward difference of at least δ, i.e., $R(L^+) - R(L^-) \geq \delta$, and the negative example violates at least one hard constraint (e.g., HASOVERLAP is true or BREAKSSEMANTICORDER is true). We define the final preference dataset as $\mathcal{D}_{\text{pref}} = \{(I, L_0, L^+, L^-) \mid \Delta R \geq \delta \text{ and } L^- \text{ violates constraint}\}$, enabling supervision-free contrastive learning that trains the model not only to follow layout instructions but also to internalize design principles such as clarity, structure, and coherence.

Direct Preference Optimization. Given preference pairs (I, L_0, L^+, L^-), we train the model using Direct Preference Optimization (Rafailov et al., 2024) objective, defined as $\mathcal{L}_{\text{DPO}}(\theta) = -\log \frac{e^{\beta \log P_\theta(L^+|I,L_0)}}{e^{\beta \log P_\theta(L^+|I,L_0)} + e^{\beta \log P_\theta(L^-|I,L_0)}}$, where β controls the sharpness of the preference signal. Unlike SFT, DPO explicitly aligns generation with structural and semantic layout quality, enabling the model to prefer globally coherent outputs in a single forward pass. During inference time, the **RewardDPO** model directly generates a high-quality layout $L^* = f_\theta(L_0, E)$ in a single forward pass, without requiring iterative refinement.

5 Experimental Setup and Main Results

To assess whether layout and content generated by LLMs/VLMs after edit instruction are "smart" enough, we benchmark model's capabilities along various dimensions. To this end, we structure our experiments as below:

5.1 Results on SMARTEditBench

We evaluate the 'smartness' of LLMs and VLMs to execute layout and semantic content editing in line with human design principles. Our experiments include three LLMs—**Gemma**, **LLaMA** (Touvron et al., 2023), and **Qwen** (Bai et al., 2023)—and three VLMs—**GPT-4o** (et al., 2024b), **Gemini-Pro** (et al., 2024a), and **LLaVA** (Liu et al., 2023)—each evaluated under four inference settings: **Zero-Shot (ZS)**, **Few-Shot (FS)**, **Chain-of-Thought (CoT)**, and **SMART-Editor Reward-Refine (SE)** (Appendix 12.4 for prompts, inference settings and HuggingFace-Models). All three LLMs and three VLMs generate layout edits in the SMART-Editor's Reward-Refine variant across all domains, while we use LLMs onlyy in RewardDPO fine-tuning setup (Finetuning details in Appendix 12.5). When the instruction changes (i.e., action: `updatetext`), we delegate content generation exclusively by the target LLM, which identifies target regions and rewrites content for specific sections accordingly. We have used LayoutPrompter (Shi et al., 2025) as training-free baseline to compare the smartness of edits when evaluating on the human-created posters and websites.

Evaluation Criteria. We evaluate edited poster/websites using both **reference-free** and **reference-based** metrics. Reference-free evaluation includes Edit Adherence (EA), indicating faithfulness to the instruction, Narrative Coherence (NC) indicating the logical section order, Cross-Sectional Consistency (CSC) which indicates semantic alignment across all the related content, Overlap Penalty (Overlap) which rewards avoiding the element collisions, Whitespace Penalty indicating efficient space use, and Alignment Reward (Algn) indicating grid-based alignment, without requiring a gold-standard reference (details in the Appendix 12.1). Reference-based evaluation measures Semantic Consistency and Layout Similarity against human-edited images (Appendix 12.2).

Reference-Free Takeaways. Compared to LayoutPrompter, **GPT-4o-SE** shows the highest overall improvements—**+15% in Narrative Coherence**, and **+12% in Cross-Sectional Consistency**, while reducing **Whitespace (–35%)** and **Overlap (–25%)**, shifting towards spatially optimized layouts. **Gemini-Pro-SE** follows closely with gains in **Edit Adherence (+8%)**, **Alignment (+10%)**, and

Model	Avg. Semantic Consistency (1–5) ↑	Avg. Layout Similarity (1–5) ↑
LayoutPrompter (Lin et al., 2023)	—	4.01 (±0.00)
LLaMA (Zero-Shot)	4.14	3.90 (-0.11)
LLaMA + Reward-Refine	4.00	4.05 (+0.04)
LLaMA + RewardDPO	4.12	4.18 (+0.17)
Gemma (Zero-Shot)	4.10	3.78 (-0.23)
Gemma + Reward-Refine	3.88	3.93 (-0.08)
Gemma + RewardDPO	4.01	4.10 (+0.09)
Qwen (Zero-Shot)	3.11	3.97 (-0.04)
Qwen + Reward-Refine	3.88	4.03 (+0.02)
Qwen + RewardDPO	4.01	4.10 (+0.09)
LLaVA (Zero-Shot)	4.00	3.55 (-0.46)
LLaVA + Reward-Refine	3.75	3.72 (-0.29)
Gemini-Pro (Zero-Shot)	4.00	3.78 (-0.23)
Gemini-Pro + Reward-Refine	4.30	4.38 (+0.37)
GPT-4o (Zero-Shot)	**4.52**	3.98 (-0.03)
GPT-4o + Reward-Refine	**4.38**	**4.47 (+0.46)**

Table 1: Reference-based evaluation on SMARTEditBench-Posters using human-edited posters as the gold reference. We report the average GPT-4o-provided scores for Semantic Consistency (SC) and Layout Similarity (LS), with deltas showing improvement over the LayoutPrompter. Best results are highlighted in green; second-best in blue. *Reward-guided refinement consistently improves layout similarity across models, and GPT-4o with Reward-Refine as the winner, indicating that inference-time reward optimization can match or exceed training-time alignment for compositional layout editing.*

Narrative Coherence (+10%), beating ZS and FS but leaving slightly higher whitespace, reflecting a trade-off between density and legibility. **LLaMA-SE** excels structurally, showing the strongest **Alignment improvement (+20%)** and **Cross-Sectional gains (+10%)** over both LPrompter and CoT, indicating better logical ordering across layout components. Gemma-SE and Qwen-SE improve (EA +5–7% and Alignment +8–10%) over both LPrompter and zero/few-shot baselines. Overall, SE consistently surpasses all prompt-only variants in **SMARTEditBench-Websites** as well.

Reference-Based Takeaways. On **SMARTEdit-Bench-Posters**, GPT-4o + Reward-Refine has the highest layout similarity (4.47) and strong semantic consistency (4.38), beating LayoutPrompter (4.01). RewardDPO fine-tuning also boosts LLaMA and Gemma by +0.22 in layout alignment (Table 14). On **SMARTEdit-Bench-Websites**, GPT-4o + Reward-Refine again leads with LS = 4.45 (+0.45) and SC = 4.50. Gemini-Pro shows consistent gains, highlighting the benefit of reward-based refinement (Table 13).

Reward-Refine vs RewardDPO. We compare RewardDPO with Zero-Shot, Supervised Finetuning (SFT), and Reward-Refine (Figure 6) for all the LLMs in **SMART-Editor**. Across all models and domains, RewardDPO obtains the highest composite rewards, indicating better alignment with semantic and visual-spatial layout goals. While Reward-Refine improves quality through iterative correction, RewardDPO matches or exceeds its final performance with greater efficiency.

Model A vs. B	Posters	Websites
RewardDPO vs. LayoutPrompter	65.4%	72.1%
RewardDPO vs. Reward-Refine	**59.5%**	**63.3%**
RewardDPO vs. Zero-Shot	**75.5%**	**83.7%**
Reward-Refine vs. LayoutPrompter	63.2%	67.5%
Reward-Refine vs. Zero-Shot	84.7%	87.0%

Table 2: Percentage of cases where one model's edits were preferred over another based on Win-Rates by Humans. *Across both* Posters *and Websites, LLama-**Reward-DPO** has the highest preference rates over all baselines.*

Ablation	EA ↑	NC ↑	Overlap ↓	Algn ↑
Reward-Refine (Full)	4.40	0.41	-62.0	33.2
A1: — Reward Replanning	4.22	0.35	-71.0	30.0
A2: — Language Feedback	4.28	0.33	-66.5	31.2
RewardDPO (Full)	4.55	0.44	-58.0	34.0
A3: — Rand Negatives (RandNeg)	4.42	0.37	-66.0	31.5
A4: — No Reward Filtering	4.40	0.39	-63.5	32.0

Table 3: Ablation results showing *both Reward-Refine and RewardDPO rely heavily on reward-driven structure removing reward-based re-planning or curated preference data harms performance.*

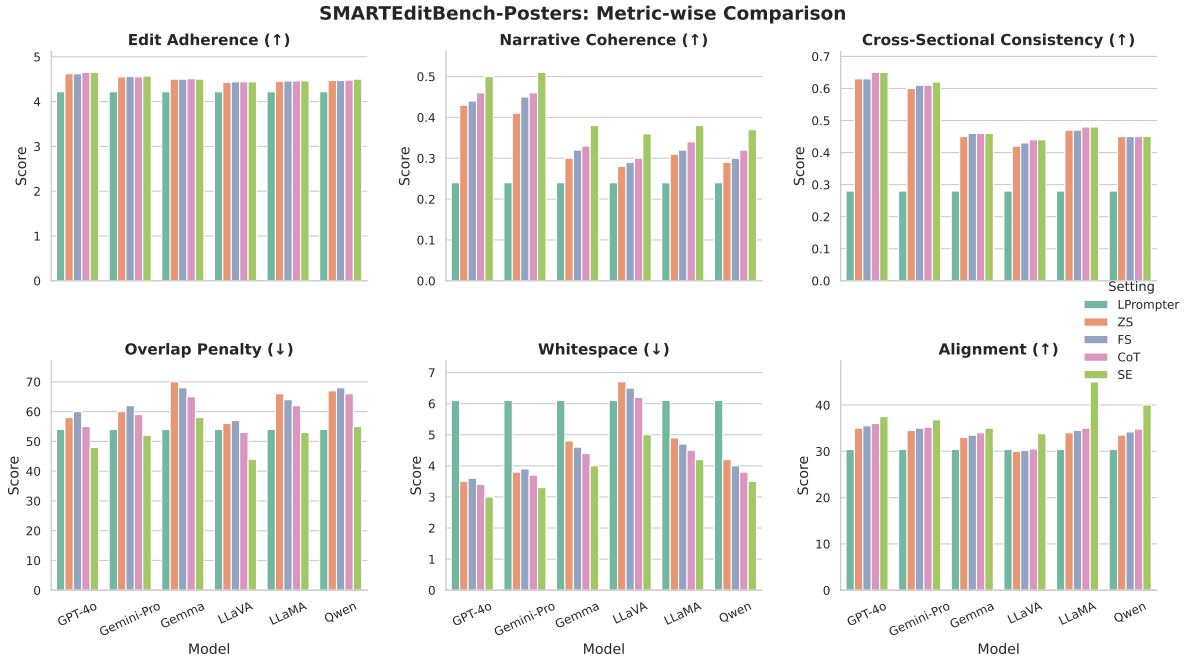

Figure 3: Comparison of model performance on SMARTEditBench-Posters across six normalized metrics. SE (SMART-Editor Reward-Refine) obtains the best overall balance, while LayoutPrompter (Lin et al., 2023) trails on structural metrics. *Edit Adherence and Cross-Sectional Consistency remain largely stable across settings, while SE shows huge improvements on Narrative Coherence, WhiteSpace Management and Alignment, suggesting that existing models follow edits blindly but lack global structural and compositional reasoning.*

5.2 Further Analysis in SMART-Editor

We have done further analysis to understand how much important are the various components in SMART-Editor.

Analysis of Iterations in Reward-Refine. To assess how well models improve layout and content over time, we apply **Reward-Refine**, iteratively updating model outputs using composite reward feedback as summation of metric scores (Details in Appendix 12.3). Each round optimizes two key reasoning dimensions: semantic consistency and visual-spatial layout. All models improve **visual–spatial layout rewards** sharply, while **semantic consistency** improves slightly, suggesting that the refinement loop optimizes compositional quality rather than deeper semantic alignment (Figure 4).

Analysis of Components in SMART-Editor. To better understand the contribution of individual components, we ablate (Table 3) and each ablation selectively removes a key mechanism from either Reward-Refine or RewardDPO.

A1: No Reward-Based Replanning (Reward-Refine) We disable iterative reward-guided refinement and use only initial layout edits to test whether passive edits suffice; this leads to a 9–11 point drop in layout quality: structured edits require active spatial reasoning.

A2: No Language Feedback (Reward-Refine) We remove intermediate natural language feedback prompts to assess their role in guiding semantic fixes: semantic consistency degrades without localized corrective signals.

A3: Random Negatives (RewardDPO) We substitute structured corrupted pairs with random preferences during DPO training: this weakens narrative and alignment understanding, resulting in incoherent section ordering and poor layout structure.

A4: No Reward Filtering (RewardDPO) We skip filtering preference pairs based on reward alignment to see how noise affects preference modeling; this introduces contradictory supervision, reducing the model's ability to align edits with the intended quality axes. Additionally we ablate the use of training data in RewardDPO and the use of critics (Appendix 13.1, 13.2, 13.3).

Analysis of Edit Types. To pinpoint specific weaknesses in model reasoning, we analyze the model behavior under four common edit operations—*Insert*, *Delete*, *Reordering*, and *Content*

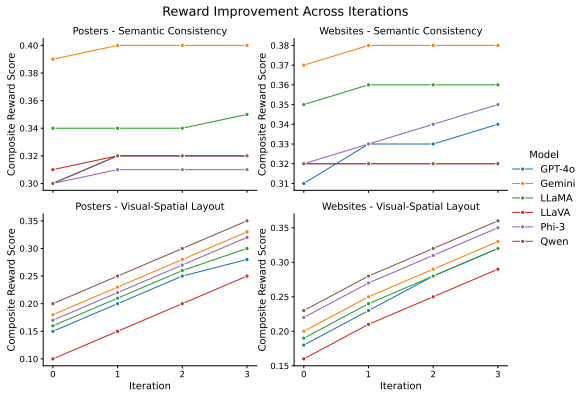

Figure 4: Cumulative reward trends in Reward-Refine (SE) *show minimal gains in semantic consistency across iterations, while visual–spatial rewards improve sharply.*

Grouping—by measuring reward deltas between pre-edit and post-edits. Layout reward is most sensitive, with severe degradation from *Insert* (-17.6%) and *Reordering* (-15.9%) edits due to increased overlap and whitespace (Table 7). Narrative coherence also drops sharply under *Insert* (-12.3%) and *Delete* (-10.0%) due to unbalanced or contextless updates. In contrast, *Cross-Sectional Consistency* is minimally affected (-0.22%), suggesting models better preserve high-level structural alignment than local spatial or narrative coherence.

5.3 Cost-Performance Trade-off Analysis

While the Zero-shot (ZS) GPT-4o (Actor) requires approximately 10.4 seconds per sample for inference, the SMART-Editor Reward-Refine (SE) approach takes about 32 seconds per sample (Table 6). Despite the higher runtime, SE consistently delivers positive improvements across all reference-free reward dimensions, with an average gain of 10.3% over ZS. We also compare SE against a simpler zero-shot GPT-4o setup that uses only the image and edit instruction as input; this baseline runs in roughly two seconds per sample but with substantially lower reward scores, with edit adherence dropping below 3.8 out of 5. Compared to this faster baseline, SE gets a 22% improvement in overall rewards. RewardDPO provides a middle ground, with an inference time of about four seconds and a performance gain of 12%, though it requires additional overhead of fine-tuning.

5.4 Human Evaluation

We conduct a human evaluation on 60 edit instances, each annotated by three independent raters, comparing four model variants: Zero-Shot, LayoutPrompter (Lin et al., 2023), Reward-Refine, and RewardDPO versions of LLaMA, and we ask annotators to rank the edited outputs based on overall edit quality given the same instruction and initial layout, considering factors such as layout compactness, alignment, and preservation of narrative flow.

Both reward-guided variants are consistently preferred over training-free baselines in both *Posters* and *Websites* (Table 2). Reward-Refine wins over Zero-Shot in 84.7% of poster edits and 87.0% of website edits, indicating that inference-time reward optimization alone leads to large perceptual gains. RewardDPO further improves over LayoutPrompter and Zero-Shot with a to 83.7% win-rate on websites. While RewardDPO is also favored over Reward-Refine (59.5% on posters and 63.3% on websites), the gap is smaller than the improvement over non-reward baselines, suggesting that both inference-time and training-time reward alignment contribute meaningfully to human-perceived edit quality.

5.5 Qualitative Case Studies

Across the case studies (Table 4), GPT-4o baseline outputs often leave posters visually cluttered, with frequent section overlaps (High Overlap Penalty), underutilized canvas space (High WhiteSpace Penalty), and misaligned boundaries that break grid structure (Low Alignment). In contrast, SMART-Editor (Reward-Refine) consistently reduces overlaps (Figure 10), redistributes whitespace to use the canvas more effectively (Figure 7, 8, 9), and aligns sections cleanly along grid lines, producing layouts that are easier to scan and semantically more coherent (High Narrative Coherence). For deletions (Figure 7, 9), SMART-Editor goes beyond removing content (e.g., Research Questions) by reshuffling surrounding sections to maintain compactness and logical structure, unlike the baseline which leaves large unused gaps. RewardDPO also resolves residual overlaps and re-establishes alignment after new sections are introduced into scientific posters (Figure 13).

6 Related Work

Our work lies at the intersection of instruction-based editing, structured layout modeling, and compositional reasoning. Instruction-guided image editing approaches like InstructPix2Pix (Brooks et al., 2023), HumanEdit (Bai et al., 2025), DIY (Jamil et al., 2025), R-SIE (Jung et al., 2025)

Edit Instruction	Figures	Comments on Baseline Edited Image (Left)	Comments on SMART-EDITOR Edited Image (Right)
Remove the abstract of the poster	Fig. 7	GPT-4o (ZS baseline) had higher penalties: Overlap penalty of 0, Wspc penalty of 0.35, and alignment reward only about 0.55.	SMART-Editor clearly improved: Ovlp same as 0, Wspc tightened to 0.12, and alignment reward increased to 0.78.
Add a Results Subsection inside the poster	Fig. 8	GPT-4o (ZS baseline) shows higher penalties on Whitespace, and alignment reward of 0.52. However, adding the Results subsection after the Conclusion breaks the natural top-down reading order, so narrative coherence drop	SMART-Editor improves whitespace while boosting alignment reward and narrative coherence by putting the summary section below.
Delete Research Questions Section	Fig. 9	GPT-4o handles the deletion of the Research Questions section poorly, leaving excessive whitespace and disrupting balance, which penalizes compactness and readability	SMART-Editor not only deletes the section but also reshuffles and realigns the remaining content, reducing wasted space and improving visual structure.
Insert a video section at the middle	Fig. 10	The video section is dropped into the canvas without much adjustment. This causes nearby sections to collide and leaves uneven whitespace around it.	The video section is inserted more thoughtfully, spacing is redistributed, overlaps are minimized, and sections align better along the grid.

Table 4: Qualitative case studies (all the examples can be found in the appendix) comparing baseline zero-shot GPT-4o edits (left) with SMART-Editor outputs (right) under diverse layout-editing instructions. Each row presents the applied edit instruction, the corresponding visual examples, and diagnostic commentary on layout quality, including whitespace usage, alignment, overlap, and narrative coherence. *While zero-shot edits often apply the requested change in isolation—leading to excess whitespace, misalignment, or disrupted reading order—SMART-Editor consistently does cascading, global adjustments that rebalance surrounding elements.*

and HQ-Edit (Hui et al., 2024) focus on pixel-level transformations or attention-weight manipulations in the diffusion models but largely ignore how edits propagate through structured scenes. While these models improve visual realism, they lack mechanisms to reason over cascading changes in layout or content coherence.

In the structured domain, models such as LayoutGAN (Li et al., 2019), LayoutTransformer (Inoue et al., 2023), and LayoutGPT (Feng et al., 2023) synthesize layouts from scratch but do not support iterative editing or reward-based refinement. Recent work like LayoutCoT (Shi et al., 2025) brings reasoning into layout generation but lacks evaluation of edit propagation. Benchmarks such as SciPostLayout (Tanaka et al., 2024), PosterCraft (Chen et al., 2025), PosterO (Hsu and Peng, 2025), PosterGen (Zhang et al., 2025b), CreatiPoster (Zhang et al., 2025a) and InstructEdit (Wang et al., 2023) support layout generation and understanding, but not compositional edit evaluation. Programmatic editing in code (Fried et al., 2023) and HTML (Li et al., 2022) explores instruction-following with symbolic reasoning, which inspires our approach to layout action planning. However, these systems operate in text/code domains and lack alignment with visual-semantic layout structures. The closest editing approach to us is the multi-agent framework GenArtist (Wang et al., 2024) that unifies image generation and editing via tool orchestration, but our work differs in scope and mechanism. GenArtist focuses on general-purpose pixel-level generation and editing across arbitrary scenes, using an MLLM agent to select from diverse external models. By contrast,we target layout-centric artifacts, where edits are not just visual but structural and logical (More in Appendix 10).

7 Conclusion and Future Work

We introduce SMART-EDITOR, a unified framework for instruction-guided layout and scene editing that emphasizes design integrity and semantic consistency. By combining inference-time refinement and training-time preference optimization, SMART-Editor obtains significant improvements in both structured and unstructured domains.

Looking ahead, we envision extending SMART-EDITOR to support multi-turn editing workflows, where iterative feedback and evolving goals drive layout evolution over time. Additionally, we aim to integrate human-in-the-loop capabilities that allow real users to steer and critique layout edits interactively—bridging the gap between automated layout generation and practical co-creative design tools used in the real world.

Limitations

1. **Assumption of Reliable Rewards:** Reward-Refine assumes access to well-defined, reliable reward functions. In real-world design workflows, such signals may be noisy, latent, or subjective.
2. **Single-Turn Editing Focus:** Our framework is evaluated on isolated editing instructions. Multi-turn, interactive layout editing remains unexplored and may require modeling user intent over time.
3. **Inference-Time Overhead:** Reward-Refine improves output quality but incurs additional computational cost during inference and may be sensitive to initialization or local optima.

Ethics Statement

The experiments performed in this study involved human participants. All the experiments involving human evaluation were exempt under institutional IRB review. We recruited participants for our human study using Upwork and we have fairly compensated all the Upwork freelancers involved in this study, at an average rate of 15.00 USD per hour (respecting their suggested Upwork hourly wage). We did not collect any personal data during the experiments, and they could choose not to participate in the study. The documents used in the study are distributed under CC-BY and MIT license.

While this work adheres to ethical practices for human participation and open data usage, it presents some potential risks. First, automated layout modifications—especially in scientific or web-based content—may unintentionally alter the intended communicative structure or introduce misinformation if misused. Second, systems that rely on implicit, instruction-based editing could generate plausible but incorrect edits that are difficult to verify without human oversight, potentially reducing user trust. Finally, the deployment of such editing systems in real-world design workflows without adequate guardrails may diminish creative control.

AI-Assistant Usage

The authors have used AI assistants (OpenAI's GPT4o) only for editorial and organizational support during the preparation of draft. Specifically, we used it to help with grammar correction, sentence restructuring, formatting, and in improving figure captions and tables, and occasionally to debug the coding implementation.

References

Jinbin Bai, Wei Chow, Ling Yang, Xiangtai Li, Juncheng Li, Hanwang Zhang, and Shuicheng Yan. 2025. Humanedit: A high-quality human-rewarded dataset for instruction-based image editing. *Preprint*, arXiv:2412.04280.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

SiXiang Chen, Jianyu Lai, Jialin Gao, Tian Ye, Haoyu Chen, Hengyu Shi, Shitong Shao, Yunlong Lin, Song Fei, Zhaohu Xing, Yeying Jin, Junfeng Luo, Xiaoming Wei, and Lei Zhu. 2025. PosterCraft: Rethinking high-quality aesthetic poster generation in a unified framework. *Preprint*, arXiv:2506.10741.

Gemini Team et al. 2024a. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

OpenAI et al. 2024b. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. 2023. LayoutGPT: compositional visual planning and generation with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen tau Yih, Luke Zettlemoyer, and Mike Lewis. 2023. InCoder: A generative model for code infilling and synthesis. *Preprint*, arXiv:2204.05999.

Hanan Gani, Shariq Farooq Bhat, Muzammal Naseer, Salman Khan, and Peter Wonka. 2024. LLM Blueprint: Enabling text-to-image generation with complex and detailed prompts. *Preprint*, arXiv:2310.10640.

Claire Golomb. 1987. The development of compositional strategies in children's drawings. *Visual Arts Research*, pages 42–52.

HsiaoYuan Hsu and Yuxin Peng. 2025. PosterO: Structuring layout trees to enable language models in generalized content-aware layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8117–8127.

Mude Hui, Siwei Yang, Bingchen Zhao, Yichun Shi, Heng Wang, Peng Wang, Yuyin Zhou, and Cihang Xie. 2024. HQ-Edit: A high-quality dataset for instruction-based image editing. *Preprint*, arXiv:2404.09990.

Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. 2023. LayoutDM: Discrete Diffusion Model for Controllable Layout Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10167–10176.

Sofia Jamil, Kotla Sai Charan, Sriparna Saha, Koustava Goswami, and Joseph K J. 2025. *Do It Yourself (DIY)*: Modifying images for poems in a zero-shot setting using weighted prompt manipulation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*.

Peidong Jia, Chenxuan Li, Yuhui Yuan, Zeyu Liu, Yichao Shen, Bohan Chen, Xingru Chen, Yinglin Zheng, Dong Chen, Ji Li, Xiaodong Xie, Shanghang Zhang, and Baining Guo. 2024. COLE: A hierarchical generation framework for multi-layered and editable graphic design. *Preprint*, arXiv:2311.16974.

Yeonjoon Jung, Seungtaek Choi, and Seung-won Hwang. 2025. Overcoming source object grounding for semantic image editing. *Transactions of the Association for Computational Linguistics*, 13:1171–1185.

Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. 2019. LayoutVAE: Stochastic scene layout generation from a label set. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Dayeon Ki, Tianyi Zhou, Marine Carpuat, Gang Wu, Puneet Mathur, and Viswanathan Swaminathan. 2025. GraphicBench: A planning benchmark for graphic design with language agents. *Preprint*, arXiv:2504.11571.

Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. 2019. LayoutGAN: Generating graphic layouts with wireframe discriminators. *Preprint*, arXiv:1901.06767.

Jingyu Li, Wei Liu, and Tan Lee. 2022. Editnet: A lightweight network for unsupervised domain adaptation in speaker verification. *Preprint*, arXiv:2206.07548.

Jiawei Lin, Jiaqi Guo, Shizhao Sun, Zijiang Yang, Jian-Guang Lou, and Dongmei Zhang. 2023. LayoutPrompter: Awaken the design ability of large language models. In *NeurIPS 2023*.

Yuanze Lin, Yi-Wen Chen, Yi-Hsuan Tsai, Lu Jiang, and Ming-Hsuan Yang. 2024. Text-driven image editing via learnable regions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7059–7068.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Preprint*, arXiv:2304.08485.

George A Lozano. 2014. Ethics of using language editing services in an era of digital communication and heavily multi-authored papers. *Science and Engineering Ethics*, 20(2):363–377.

Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. 2023. Drag Your GAN: Interactive point-based manipulation on the generative image manifold. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, SIGGRAPH '23, page 1–11. ACM.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. *Preprint*, arXiv:2103.00020.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct Preference Optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Hengyu Shi, Junhao Su, Huansheng Ning, Xiaoming Wei, and Jialin Gao. 2025. LayoutCoT: Unleashing the deep reasoning potential of large language models for layout generation. *Preprint*, arXiv:2504.10829.

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

Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. 2025. Design2Code: Benchmarking multimodal code generation for automated front-end engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3956–3974.

Manan Suri, Puneet Mathur, Franck Dernoncourt, Rajiv Jain, Vlad I Morariu, Ramit Sawhney, Preslav Nakov, and Dinesh Manocha. 2024. DocEdit-v2: Document structure editing via multimodal LLM grounding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15485–15505.

Shohei Tanaka, Hao Wang, and Yoshitaka Ushiku. 2024. SciPostLayout: A dataset for layout analysis and layout generation of scientific posters. *Preprint*, arXiv:2407.19787.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Qian Wang, Biao Zhang, Michael Birsak, and Peter Wonka. 2023. InstructEdit: Improving automatic masks for diffusion-based image editing with user instructions. *Preprint*, arXiv:2305.18047.

Zhenyu Wang, Aoxue Li, Zhenguo Li, and Xihui Liu. 2024. GenArtist: Multimodal llm as an agent for unified image generation and editing. *Preprint*, arXiv:2407.05600.

Shu Zhang, Xinyi Yang, Yihao Feng, Can Qin, Chia-Chih Chen, Ning Yu, Zeyuan Chen, Huan Wang, Silvio Savarese, Stefano Ermon, Caiming Xiong, and Ran Xu. 2024. HIVE: Harnessing human feedback for instructional visual editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9026–9036.

Zhao Zhang, Yutao Cheng, Dexiang Hong, Maoke Yang, Gonglei Shi, Lei Ma, Hui Zhang, Jie Shao, and Xinglong Wu. 2025a. CreatiPoster: Towards editable and controllable multi-layer graphic design generation. *Preprint*, arXiv:2506.10890.

Zhilin Zhang, Xiang Zhang, Jiaqi Wei, Yiwei Xu, and Chenyu You. 2025b. PosterGen: Aesthetic-aware paper-to-poster generation via multi-agent llms. *Preprint*, arXiv:2508.17188.

Wanrong Zhu, Ruiyi Zhang, Jennifer Healey, William Yang Wang, and Tong Sun. 2024. Automatic layout planning for visually-rich documents with instruction-following models. In *Proceedings of the 3rd Workshop on Advances in Language and Vision Research (ALVR)*, pages 167–172.

8 Appendix

We organize the appendix section into subsections to furnish additional details supporting our claims:

- Human Evaluation to derive the motivation of our research (Appendix 9)
- Related Works (Appendix 10)
- **SMARTEdit-Bench** Statistics
- **SMART-Editor** Methodology (Appendix 11)
  - Preprocessing Details of Images in **SMARTEdit-Bench** (Appendix 11.1)
  - Edit Instruction Taxonomy (Appendix 11.2)
  - Action Agent Details (Appendix 11.3)
  - Critique Agent Details (Appendix 11.4)
  - Optimizer Agent Details (Appendix 11.5)
- Implementation and Evaluation (Appendix 12)
  - Reference-Free Metrics (Appendix 12.1)
  - Reference-Based Evaluation Metrics on Structured Images (Appendix 12.2)
  - Baseline Inference Prompting and Model Details (Appendix 12.4)
  - Composite Reward for Evaluation (Appendix 12.3)
  - DPO Implementation and Hyperparameters (Appendix 12.5)
- Additional Results on Automatic Evaluation (Appendix 13)
  - Reference-Based Evaluation on Websites and Posters (Table 14, Table 13)
  - Analysis of Training Data in DPO (Appendix 13.1)
  - Analysis of Beam Search Iteration (Appendix 13.2)
  - Ablation of Reward Components in RewardRefine Approach (Appendix 13.3)

9 Human Evaluation for Motivation

To assess whether the edit plans generated by VLMs and LLMs translate into coherent and reward-aligned edits, we do a human evaluation across 30 examples in each domain (posters and websites). For each sample, we provide human

Domain	Plan-to-Execution Match (1–5)	Avg. Overlap Violation	Avg. Alignment Error
Posters	4.45 ± 0.31	1.22 ± 0.45	1.40 ± 0.51
Websites	4.18 ± 0.39	1.50 ± 0.62	1.68 ± 0.58

Table 5: Human evaluation of plan-to-execution consistency and reward function failures (rated 1–5) across 30 samples per domain by 3 expert annotators. *Posters obtain higher execution fidelity than websites, which incur more frequent spatial violations.*

annotators (3 PhD students with design expertise) with the original image/layout, the model-generated plan, and the final layout/output after plan-to-command execution.

Annotators rate: (i) how well the plan aligns with the observed edit (Plan-to-Execution Match, 1–5), and (ii) the degree of overlap and alignment errors in the final layout (1–5, lower is worse). Posters show the highest alignment (avg. 4.45), with minimal overlap or misalignment (Table 5). Websites do slightly worse (4.18).

These results highlight a key gap in current models: while structured edits propagate well in posters and web layouts, plan-grounded editing in natural images is hindered by poor affordance reasoning and visual grounding. Importantly, failures identified by annotators—such as "misplaced added sections," "overlap with titles," or "semantic drift in relocated content"—correlate with quantitative penalties from our reward functions.

Setup. We selected 30 representative edit examples per domain. For each, we showed three expert annotators: (a) input layout or image, (b) edit instruction, (c) plan generated by the VLM/LLM, and (d) final output after applying the generated plan through command and execution steps.

Metrics. Annotators rated (1) how well the final output reflected the intended plan (Plan-to-Execution Match), and (2) overlap or misalignment based on visual inspection and design conventions (1–5 scale). Scores were averaged across annotators.

Participants. Three PhD-level design experts with prior experience in scientific poster creation and multimodal editing workflows.

Findings. Structured domains (posters, websites) exhibited strong plan adherence and minimal reward violations (Table 5).

10 Related Work

Our work intersects multiple areas of research, including instruction-based image editing, layout and structure-preserving generation, and cascading edit reasoning. We organize the discussion into four main categories: **(1) Instruction-Based Image Editing**, **(2) Structured Layout Editing**, **(3) Compositional and Cascading Reasoning**, and **(4) Programmatic and Code-Based Edit Modeling**.

Structured Layout Generation Structured layout generation has gained attention for tasks such as UI prototyping, poster generation, and scientific visualization. Models like **LayoutGAN** (Li et al., 2019), **LayoutVAE** (Jyothi et al., 2019), and **LayoutTransformer** (Inoue et al., 2023) learned to place elements based on supervised geometric and semantic patterns.

With the rise of LLMs, **LayoutGPT** (Feng et al., 2023) proposed using LLMs to serve as visual planners by converting textual instructions into HTML-style structured layouts. Similarly, **LayoutCoT** (Shi et al., 2025) demonstrates that CoT reasoning and retrieval-augmented generation can help LLMs improve alignment and visual coherence in layout generation. However, both target layout *synthesis*, not **iterative layout editing**, and lack evaluation mechanisms for edit propagation. **SciPostLayout** (Tanaka et al., 2024) introduces a dataset of scientific poster layouts, enabling research into layout understanding. Yet it focuses on static layout prediction and lacks instructional editing or reasoning evaluation. **InstructEdit** (Wang et al., 2023) explores using instructions to guide edits, optimizing for downstream engagement (social media) or using segmentation for precise control. These approaches are instructive for our work but again, they operate at the **pixel-level or object-level**, not layout-structure.

Compositional and Cascading Reasoning in Layouts Humans execute layout edits not just locally, but with global awareness—adding a new section may require shifting others, updating or-

Figure 5: Comparison of model performance on SMARTEditBench-Websites across six normalized metrics. SE (SMART-Editor Reward-Refine) obtains the best overall balance, while LayoutPrompter (Lin et al., 2023) trails on structural metrics. *Edit Adherence and Cross-Sectional Consistency remain largely stable across settings, while SE shows huge improvements on Narrative Coherence, WhiteSpace Management and Alignment, suggesting that existing models follow edits blindly but lack global structural and compositional reasoning.*

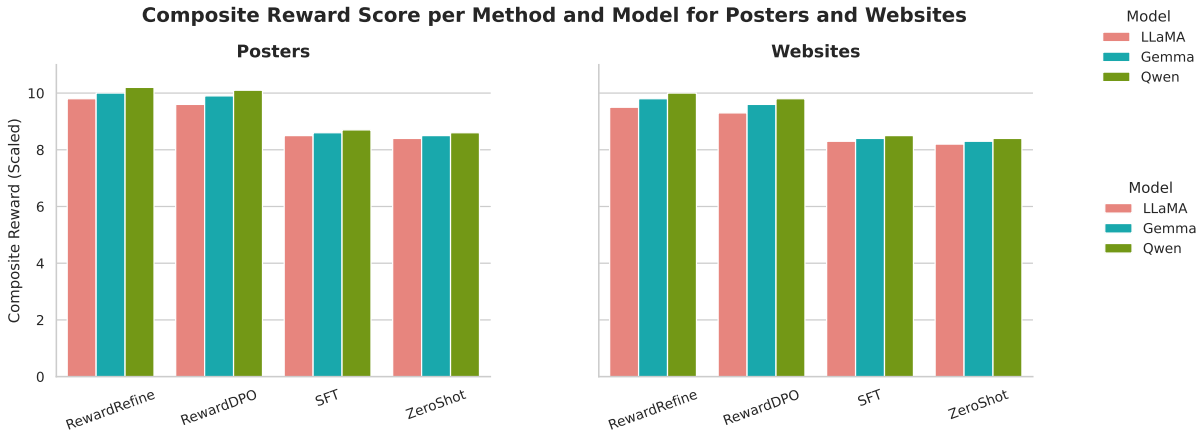

Figure 6: *Models trained with RewardDPO and refined using Reward-Refine obtain higher composite rewards—reflecting better semantic and spatial layout quality—compared to SFT and Zero-Shot baselines.*

Method	Inference Time / sample	Reward Gain (%)	Notes
Zero-Shot GPT-4o (Actor)	10.4s	—	Full-context input
Zero-Shot GPT-4o (Image + Edit only)	2s	Baseline	Edit Adherence $< 3.8/5$
SMART-Editor (Reward-Refine, SE)	32s	+10.3% vs ZS (Actor)+22% vs Img+Edit baseline	Strongest overall rewards
RewardDPO	4s	+12% vs baseline	Extra cost: finetuning + preference data

Table 6: Inference analysis comparing efficiency (time per sample) and reward improvements across Zero-Shot GPT-4o variants, SMART-Editor Reward-Refine, and RewardDPO. *While SMART-Editor Reward-Refine delivers the strongest rewards, faster alternatives such as RewardDPO may be preferable in latency-sensitive deployment settings.*

der, and maintaining alignment. Existing editing benchmarks (e.g., **GraphicBench**) (Ki et al., 2025) focus on agent planning and multimodal synthesis but do not evaluate how edits *cascade* through

Edit Type	Narrative Coherence ↓	Layout ↓	Cross-Section Consistency ↓
Insert	-12.3%	**-17.6%**	-0.22%
Delete	-10.0%	**-14.4%**	-0.12%
Reordering	-0.08%	**-15.9%**	0%
Content Grouping	-0.12%	**-12.7%**	0%

Table 7: Average relative reward drop across reasoning dimensions by edit type (from zero-shot baseline). *Layout degradation dominates across Insert, Delete, and Grouping edits, followed by narrative incoherence. Cross-section consistency is much more preserved.*

Dataset / System	Domain(s)	Edit-Aware	Comp. Reasoning	Cascading	Instruction	Eval Scope	Notable Features
InstructEdit (Wang et al., 2023)	Natural Images	✓	—	—	Natural Language	Visual, Mask Accuracy	LLM + segmentation + SD for multi-object edits.
PostScoreEdit	Social Media	✓	—	—	Prompt Paraphrasing	Engagement, Human Ratings	Paraphrase+rank to boost audience engagement.
FFCLIP (Radford et al., 2021)	Faces, Cars, Churches	✓	Partial	—	Free-form Text	CLIP Alignment	StyleGAN latent editing via CLIP alignment.
DocEdit-v2 (Suri et al., 2024)	Structured Docs	✓	Local	—	Free/Structured	DOM Tree, CSS IoU	LMM-based HTML editing + reformulation.
LayoutGPT (Feng et al., 2023)	2D/3D Layouts	—	✓	—	Prompt Only	Layout Plausibility	CSS-like layout synthesis with spatial reasoning.
COLE (Jia et al., 2024)	Graphic Design	—	Hierarchical	—	Design Intent	Visual + Typographic	JSON + image gen for typography and layout.
SciPostLayout (Tanaka et al., 2024)	Scientific Posters	—	—	—	Layout Only	Visual Structure	Poster layout generation only, no edit modeling.
LayoutCoT (Shi et al., 2025)	Posters, UI Layouts	—	CoT Only	—	CoT Prompting	Visual Coherence	Prompt-based spatial reasoning, no edit ops.
SMARTEdit-Bench (Ours)	Posters, Web, Images	✓	✓	✓	Implicit instructions requiring Cascaded-Reasoning	Visual, Semantic, Structural	Multi-domain, layout propagation, semantic + visual edit cascades.

Table 8: Comparison of SMARTEdit-Bench with prior datasets. Our benchmark uniquely supports multi-domain, instruction-grounded cascading edits with compositional reasoning. Related works are described in Appendix 10.

a layout. To this end, we introduce **SMARTEditBench**, a benchmark for testing whether multimodal LLMs can handle *compositional reasoning* in edits—e.g., does repositioning one figure preserve alignment with its caption? Can adding a section reorder surrounding elements semantically? Our method evaluates edits in structured settings like posters, web layouts, and scenes using metrics like **semantic ordering, spatial overlap, and visual alignment**. This direction is loosely inspired by structured editing tasks in other modalities, such as **semantic parsing** with structured outputs, but uniquely adapted for **visual-semantic alignment and layout reasoning**.

Programmatic Editing and Code-Driven Reasoning. Recent research investigates editing behaviors over code or structured representations. Instruction-following models for **code editing** (e.g., CodeX (Chen et al., 2021), InCoder (Fried et al., 2023)) have demonstrated impressive reasoning about insertion, deletion, and reordering in code. In a similar vein, some web-editing models like **EditNet** (Li et al., 2022) predict edit actions over HTML based on goal descriptions. In the vision domain, LayoutGPT (Feng et al., 2023) uses a stylesheet-style intermediate representation to enable compositional planning. Our work draws on these insights and applies them to layout editing, where bounding box structures and section types must be jointly updated to follow instructions while preserving document integrity. In contrast to prior work focused on localized edits, synthetic instruction adherence, or single-step generation, our framework evaluates whether multimodal models can execute **cascading, multi-element, instruction-based layout refinement**. We propose a new benchmark (*SMARTEditBench*), and evaluation protocols to test whether edits to one element propagate logically and semantically across a structured scene or document.

11 SMART-Editor Methodology

In this section, we discuss the steps of our methodology and their detailed implementation.

11.1 Preprocessing Details

To enable structured layout reasoning across diverse input formats, we apply modality-specific preprocessing techniques to convert unstructured inputs into a unified representation: a set of objects or sections, each with a bounding box and textual content. Below we describe the processing pipeline for each modality: To enable structured layout reasoning across diverse input formats, we adopt modality-specific preprocessing pipelines that convert raw visual or web-based inputs into a unified representation consisting of objects or sections with associated bounding boxes and textual content.

For **scientific posters**, we use GPT-4o to extract structured layout elements directly from poster screenshots or PDF renderings. We prompt the model with a query such as: *Given this poster, identify all distinct rectangular sections and provide their bounding boxes and associated content in the format {section name: [x1, y1, x2, y2], content:*. GPT-4o returns a structured layout with section-wise bounding boxes and corresponding textual

descriptions, enabling layout-aware downstream reasoning.

For **websites**, we parse HTML DOM tree using tools such as `BeautifulSoup` or `lxml` to extract key structural components, including headers, navigation bars, content blocks, and sidebars. We retrieve the text content for each major DOM node and estimate their bounding boxes using browser-based rendering APIs (e.g., via `Playwright` or headless browser utilities). The result is a structured mapping from each layout region to its geometric footprint and textual content, making it compatible with layout editing models.

11.2 Edit Instruction Taxonomy

Table 10 provides illustrative examples of edit instructions spanning diverse modalities (posters, websites), each annotated with a corresponding taxonomy of edit type, degree of explicitness, and need for cascaded changes. We distinguish between five types of structural edits: **Section Reordering**, **Section Insertion**, **Object Replacement**, **Content Grouping**, and **Section Deletion**. These edit types are frequently encountered across structured and unstructured domains in our benchmark. An important dimension captured in this table is **implicitness**—i.e., how much of the required layout or content adjustment is left unstated in the instruction. For instance, even simple insertions (e.g., adding a section) demand implicit reasoning about placement and formatting. We also highlight whether edits cause **cascaded changes**—that is, whether making the instructed change requires updates to surrounding layout structure, alignment, or spacing. Most implicit instructions lead to cascaded layout effects, making them particularly challenging for models that rely on direct decoding without structural reasoning. This taxonomy informs our design of evaluation settings and is central to motivating the need for structured, multi-step editing pipelines like SMART-Editor.

11.3 Action Agent Details

The action agents takes in the layout and content extracted from an image, translates the implicitly specified edit instruction in the layout-content space into verbal action plan, next the actions are transformed into specific modular functions (Description of modular functions in 11 and the implementation of each of these functions are in ALgorithm 2, ALgorithm 3, ALgorithm 4 and ALgorithm 5) which when executed should lead to the refined layout-content output.

Prompt: Action Plan Generation

You are provided with a structured layout consisting of bounding boxes and their associated section labels, along with an edit instruction that specifies how the layout should be changed.
Your task is to generate a clear and concise step-by-step **action plan in natural language** that explains:

- Which specific sections need to be added, removed, or modified.
- What geometric adjustments are required, such as moving, resizing, or swapping sections.
- The reasoning behind these changes, ensuring that layout coherence is preserved—for example, avoiding overlaps or maintaining narrative flow.

Edit Instruction:
`{edit_instruction}` **Initial Layout:**

`{initial_layout}`
(Each entry includes a section ID, bounding box coordinates $[x_0, y_0, x_1, y_1]$, and a section label.)
Expected Output:
Write a natural language action plan describing the semantic and spatial edits needed to fulfill the instruction. Avoid code or structured formats—this should be a verbal plan intended for a layout editing agent to interpret and execute.

Target Action Driven Action-Plan

You will be provided with an action plan corresponding to an edit instruction `{edit_instruction}` and a set of target geometric actions. Your job is to convert the plan into steps of geometric actions that would be integrated into the code sequentially.
Action Plan (in words)
`{Action_plan}`
Geometric Action Codes
`{geometric_actions}`
Initial Layout
`{initial_layout}`
Output Format
Format your output as a list of JSON objects containing the sequence of target action APIs by filling in the real values of parameters inside the functions. Do not output anything else. The output should match the exact format in the Geometric Action Codes, for example: `snap_to_grid(layout: Layout, section_id: str, grid_unit)` `resize_section(layout: Layout, section_id, scale_x, scale_y)` Each output JSON object must contain two keys: `action` and `parameters`.

11.4 Critique Agent Details

To evaluate structured document layouts such as posters and websites, we design critique agents that assess both spatial and semantic alignment

Domain	#Edits	Source	Annotations
Webpages (Real)	564	GitHub + GPT-4V	Edit Type + Reasoning (Gold Edited Output)
Webpages (Synthetic)	510	Design2Code + GPT-4o	Edit Type + Reasoning (Gold Edited Output)
Posters (Synthetic)	1,200	SciPostLayout + GPT-4o	Edit Type + Reasoning (No Gold Edited Output)
Posters (Human Eval)	250	Expert-Tagged Posters	Gold Edited Output

Table 9: Composition of SMARTEditBench across domains, data sources, and annotation types. *The benchmark spans diverse editing regimes, combining fully supervised real and synthetic webpage edits with large-scale synthetic poster edits and expert-annotated evaluation data, enabling both training and rigorous compositional evaluation.*

Edit Instruction	Taxonomy	Modality	How is it Implicit?	Cascaded Change?
Move the Results section above Methods	Section Reordering	Poster	Requires understanding of scientific discourse order (Results should follow Methods)	Yes
Add a section describing ongoing research projects below the About section	Section Insertion	Website	Implies structural addition and reflowing of layout without stating exact position or formatting	Yes
Group all job experiences by company to make the layout more readable	Content Grouping	Website	Suggests structural reorganization without detailing how the layout should shift	Yes
Remove the References section	Section Deletion	Poster	Directly states which section to delete, but does not specify consequences on layout alignment or spacing	Yes

Table 10: Examples of edit instructions with associated taxonomy, modality, nature of implicit reasoning, and whether the edit requires cascaded changes. *Many real-world edit requests underspecify layout operations, requiring models to infer structural intent and propagate changes across multiple dependent elements.*

Function	Description	Example	LaTeX Symbolic Form
`translate_object`	Shifts a section's position by Δx, Δy.	Translate `METHODS` by $(0, +100)$	$\text{bbox}' = \text{bbox} + (\Delta x, \Delta y)$
`resize_object`	Scales the width and height of a section by s_x, s_y.	Resize `PLOT 1` by $(s_x = 1.2, s_y = 0.8)$	$\text{bbox}' = \text{scale}(\text{bbox}, s_x, s_y)$
`align_object_left`	Aligns the left edge of a section to a reference x.	Align `BACKGROUND` to $x = 30$	$x_1' = x_{\text{ref}},\ x_2' = x_{\text{ref}} + w$
`center_object`	Centers the section horizontally within canvas width W.	Center `ACKNOWLEDGEMENTS` on canvas of width 1080	$x_1' = \frac{W-w}{2},\ x_2' = x_1' + w$
`snap_to_grid`	Snaps bbox edges to nearest grid unit δ.	Snap `REFERENCES` to grid of 20px	$x_i' = \delta \cdot \lfloor \frac{x_i}{\delta} + 0.5 \rfloor$
`insert_object`	Inserts a new section with given bbox and text.	Insert `SUMMARY` at $[100, 500, 400, 580]$	$\text{layout}[s] = \{\text{bbox}, \text{text}\}$
`remove_object`	Deletes a section from layout by ID.	Remove `INTRODUCTION`	$\text{layout} = \text{layout} \setminus \{s\}$
`replace_object_bbox`	Replaces the bbox of a section with a new one.	Replace `METHODS` bbox to $[50, 400, 480, 500]$	$\text{layout}[s].\text{bbox} \leftarrow \text{new\_bbox}$
`reorder_objects`	Reorders sections top-down with optional spacing.	Reorder [`INTRO`, `METHOD`, `RESULT`] with $y_{\text{pad}} = 20$	$y_i' = y_0 + \sum_{j=1}^{i-1}(h_j + y_{\text{pad}})$
`update_object_text`	Updates the text content of a section.	Update `RESULT` text to "New findings here."	$\text{layout}[s].\text{text} \leftarrow \text{new\_text}$

Table 11: Layout editing functions with their semantic descriptions, practical examples, and symbolic representations. All the implementation algorithms are listed in Algorithm 2, Algorithm 3, Algorithm 4 and Algorithm 5. *Complex layout edits can be decomposed into some interpretable, composable geometric and content-level operations, enabling precise reasoning, verification, and cascaded refinement.*

of layout edits with the instructional goal. Each agent receives the edit trace (instruction, initial layout, final layout), computed reward scores, and generates targeted, actionable feedback.

Critique Dimensions Each critique agent is responsible for one of the following layout quality metrics as calculated in Appendix 12.1. Each agent is provided with the following:

- **Edit Instruction**: Natural language instruction (e.g., "Move Results above Methods")
- **Layout JSONs**: Initial and edited layouts with bounding boxes and content
- **Reward Values**: Pre-computed scores from layout evaluators (e.g., overlap, alignment)

Each critique agent is prompted as:

```
You are a critique agent analyzing a layout edit. Instruction: Add a section titled User Study below References Initial Layout: – Final Layout: – Reward Scores:
  • Edit Adherence: 4
  • Overlap Penalty: 0.6
  • Whitespace: 0.35
  • Alignment: 0.67
  • Narrative Coherence: 0.8
  • Cross-Sectional Consistency: 1.0
For each dimension, write: 1. What is good or bad 2. What causes the issue 3. How to revise the layout to improve it
```

Agents return structured rationales (e.g., in JSON or Markdown) identifying violations and suggesting layout edits such as repositioning, snapping to grid, or restructuring slide order. These are used downstream by the Optimizer Agent.

Composite Reward and Feedback Trace. The composite reward for an unstructured image is computed by summing the individual dimension scores. Critique agents trace the reward breakdown to pinpoint low-performing aspects, providing a transparent rationale for model evaluation and improvement.

11.5 Optimizer Agent Details

12 Implementation and Evaluation Details

12.1 Reference-Free Evaluation

We have used the following reference-free metrics/rewards to calculate which do not assume any gold-standard image to compare.

1. Edit Adherence (EA) Evaluation

You are given a layout editing task. Based on the **Edit Instruction**, the system updates the layout and content. You are asked to rate how faithfully the instruction was followed.
Edit Instruction: `Insert instruction here`

Initial Layout and Content:
`Insert JSON-like structure for layout and section content before edit`

Final Layout and Content:
`Insert JSON-like structure for layout and section content after edit`

Question: On a scale from 1 to 5, how well does the final layout implement the edit instruction?

1: Edit is incorrect or missing.
3: Edit partially follows the instruction.
5: Edit fully implements the instruction as intended.

Your Response:
Score (1–5): _____ Justification:

Narrative Coherence (Narr.) Evaluation. To evaluate the **Narrative Coherence (Narr.)** (r_{sem}) evaluates whether the edited layout preserves the top-down logical structure of a document (e.g., *Background* $\rightarrow$ *Methods* $\rightarrow$ *Results*). For each pair of expected section orders (S_i, S_j), GPT-4o is prompted with: *"Does section S_i logically precede section S_j?"*. A violation is counted if the answer is negative. The final reward is computed as

$$r_{\text{sem}} = 1 - \frac{v}{|\mathcal{O}|}$$

where v is the number of violations and $\mathcal{O}$ is the set of all ordered section pairs. This reward ensures that edits do not disrupt the logical reading flow of the content (Refer to Examples in Table 12).

Cross-Sectional Consistency Evaluation. (r_{cross}) measures whether semantically linked content—such as figures and their captions or methods and results—remain meaningfully consistent. GPT-4o is treated as a natural language

Metric	Description (GPT-4o-Inferred)
Narrative Coherence	Assesses whether the document follows a top-down logical structure (e.g., Background → Methods → Results). For each section pair (S_i, S_j) in the expected order, GPT-4o is prompted with: "Does the content of section 'S_i' logically precede section 'S_j'?". If GPT-4o responds no, the pair is marked as a violation. The final reward is computed as $r_{\text{sem}} = 1 - \frac{v}{\lvert\mathcal{O}\rvert}$, where v is the number of violations and $\mathcal{O}$ is the set of expected pairs.
Cross-Sectional Consistency	Measures semantic contradictions across grouped section pairs (e.g., Figure and its Caption, or Methods and Results). GPT-4o is used as an NLI model: section S_i serves as the premise, S_j as the hypothesis. It is prompted to determine whether the pair is entailed, neutral, or contradictory. If the label is "contradiction" it is counted as a violation. We define the reward as $r_{\text{cross}} = 1 - \frac{c}{\lvert\mathcal{G}\rvert}$, where c is the number of contradictions and $\mathcal{G}$ is the set of grouped section pairs.

Table 12: Description of semantic reward metrics inferred using GPT-4o. *Semantic reward functions derived from LLM reasoning provide a practical signal for guiding and diagnosing compositional layout and content edits.*

inference model, where each section pair (S_i, S_j) is evaluated for entailment. If GPT-4o detects a semantic contradiction, it is counted as a violation. We define the reward as

$$r_{\text{cross}} = 1 - \frac{c}{|\mathcal{G}|}$$

where c is the number of contradictions and $\mathcal{G}$ is the set of grouped section pairs. This metric ensures semantic alignment across logically related content blocks (Refer to Examples in Table 12).

Overlap Penalty (Ovlp). It quantifies undesirable visual collisions between elements. For every pair of layout components $(\mathbf{b}_i, \mathbf{b}_j)$, the intersection-over-union (IoU) is computed if they overlap, and accumulated as a penalty. This discourages any spatial overlap that could hinder readability or visual clarity.

Whitespace Penalty (Wspc). It inefficient use of layout space.

$$1 - \frac{A_{\text{used}}}{A_{\text{canvas}}}$$

where A_{used} is the total area of layout boxes and A_{canvas} is the total canvas area. This encourages layouts to use space compactly, without excessive gaps that reduce visual coherence.

Alignment Reward (Algn). It encourages the visually aligned layouts by rewarding sections whose left edges fall on predefined grid lines. Let x_i be the left edge of bounding box $\mathbf{b}_i$. Then, the alignment reward is computed as

$$r_{\text{align}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}(x_i \bmod \text{grid} = 0)$$

where N is the total number of layout components. This promotes clean and structured alignment across the layout.

12.2 Reference-Based Evaluation Prompts on Structured Domains

In this section, we discuss about the different evaluation metrics that we use to compare the model-edited images with the human-edited images (gold-standard) on the structured domains of images such as posters and websites **SMARTEdit-Bench**-Posters and **SMARTEdit-Bench**-Websites. To evaluate, we use the following prompts using GPT4-o as Judge as mentioned below, and the results are tabulated in Table 14 and Table 13.

Semantic Consistency (Reference-Based)

Compare the model-edited layout and content with a human-edited (gold) version. **Instruction:** Move Results above Methods
Model Output: —
Gold Output: — **Task:** Rate semantic consistency on a scale from 1 (completely misaligned) to 5 (semantically faithful to the gold).

Layout Similarity (Reference-Based)

Compare the spatial arrangement and section order of the model-edited layout to the gold layout.
Ignore text content; focus only on structural similarity.
Model Layout: —
Gold Layout: —

Model	Avg. SC (1–5) ↑	Avg. LS (1–5) ↑
LayoutPrompter (Lin et al., 2023)	—	4.00 (±0.00)
LLaMA (ZS)	4.08	3.88 (-0.12)
LLaMA + Reward-Refine	4.05	4.10 (+0.10)
LLaMA + RewardDPO	4.18	4.22 (+0.22)
Gemma (ZS)	4.02	3.75 (-0.25)
Gemma + Reward-Refine	3.90	4.05 (+0.05)
LLaMA + RewardDPO	4.08	4.22 (+0.22)
LLaVA (ZS)	3.95	3.60 (-0.40)
LLaVA + Reward-Refine	3.85	3.80 (-0.20)
Gemini-Pro (ZS)	4.38	4.20 (+0.20)
Gemini-Pro + Reward-Refine	4.32	4.36 (+0.36)
GPT-4o (ZS)	4.45	4.30 (+0.30)
GPT-4o + Reward-Refine	**4.50**	**4.45 (+0.45)**

Table 13: Reference-based evaluation on **SMARTEditBench-Websites**, comparing Zero-Shot, Reward-Refine, and RewardDPO-enhanced variants. We report average scores (1–5) for *Semantic Consistency (SC)* and *Layout Similarity (LS)*, with deltas from the **LayoutPrompter** baseline (LS = 4.00) shown in parentheses. *Reward-guided optimization consistently improves layout similarity across model families, with GPT-4o + Reward-Refine achieving the strongest combined semantic and layout alignment.*

Rate similarity from 1 (very different) to 5 (nearly identical).

12.3 Composite Reward Calculation for Structured and Unstructured Images

In the case of posters and websites (structured), the **Composite Reward** used in Figure 6 combines all the rewards and penalties mentioned in Section 12.1.

12.4 Baseline Inference Prompting and Model Details

Zero-Shot Prompt: Layout Inference

You are a layout editing assistant. Your task is to update the structure of a poster or webpage layout in response to a human instruction. **Instruction:** `Move the Results section above Methods`
Initial Layout: `{Initial_layout}`
Initial Content: `{Initial_Content}`
Output: Return the updated layout as a JSON dictionary. Do not explain your answer.

Chain-of-Thought Prompt: Step-by-Step Layout Reasoning

You are a layout editor that follows human design principles. Think step-by-step to apply the edit instruction to the layout.
Instruction: `Insert a new section titled Discussion below Results`
Initial Layout: `{Initial_layout}`
Step-by-Step: 1. The instruction asks to insert "Discussion" below "Results".
2. "Results" ends at y=500.
3. Allocate a height of 100 for the new section.
4. Add new bounding box for "Discussion": `[0, 500, 768, 600]`
Updated Layout: `{Updated_layout}`

Few-Shot Prompt: Layout Editing with Examples

You are a layout editing assistant. Your task is to revise the layout structure based on edit instructions, layout geometry, and section content.
— Example 1 —

— Example 2 —

— Now You Try —
Instruction: `Swap Methods and Study Participants`
Initial Layout: `{Initial_layout}`
Initial Content: `{Initial_Content}`
Output: Provide updated layout only.

Inference Settings

Default Generation Configuration. Unless otherwise specified, models are run with the following decoding settings: `max_new_tokens = 1024`, `temperature = 0.7`, `top_p = 0.9`, `top_k = 50`, `repetition_penalty = 1.1`, `do_sample = True`, `num_beams = 1`.

Special Use Cases. For REWARD-REFINE, we enable `num_beams = 5` to allow multi-candidate sampling. For VLMs such as LLaVA or GPT-4o, structured visual layouts are encoded as textual descriptions of bounding boxes and section names when images are not directly available.

Model	Avg. SC (1–5) ↑	Avg. LS (1–5) ↑
LayoutPrompter (Lin et al., 2023)	—	4.01 (±0.00)
LLaMA (Zero-Shot)	4.14	3.90 (-0.11)
LLaMA + Reward-Refine	4.00	4.05 (+0.04)
LLaMA + RewardDPO	4.12	4.18 (+0.17)
Gemma (Zero-Shot)	4.10	3.78 (-0.23)
Gemma + Reward-Refine	3.88	3.93 (-0.08)
Gemma + RewardDPO	4.01	4.10 (+0.09)
Qwen (Zero-Shot)	3.11	3.97 (-0.04)
Qwen + Reward-Refine	3.88	4.03 (+0.02)
Qwen + RewardDPO	4.01	4.10 (+0.09)
LLaVA (Zero-Shot)	4.00	3.55 (-0.46)
LLaVA + Reward-Refine	3.75	3.72 (-0.29)
Gemini-Pro (Zero-Shot)	4.00	3.78 (-0.23)
Gemini-Pro + Reward-Refine	4.30	4.38 (+0.37)
GPT-4o (Zero-Shot)	4.52	3.98 (-0.03)
GPT-4o + Reward-Refine	**4.38**	**4.47 (+0.46)**

Table 14: Reference-based evaluation on **SMARTEditBench-Posters**, comparing **Zero-Shot**, **Reward-Refine**, and **RewardDPO** variants against the training-free baseline **LayoutPrompter**, using human-edited posters as the gold standard. We report average GPT-4o scores (1–5) for *Semantic Consistency (SC)* and *Layout Similarity (LS)*, with deltas indicating improvements over the LayoutPrompter baseline (SC = 4.01). *Reward-guided optimization substantially improves alignment with human-edited poster layouts, with the largest gains observed for stronger base models such as GPT-4o and Gemini-Pro.*

12.5 DPO Implementation and Hyperparameters

For DPO training, we fine-tuned both `LLaMA-3-8B-Instruct` and `Gemma-2-9B-It` using full-parameter tuning across all layers. We employed a sigmoid-based preference loss with $\beta = 0.1$, a learning rate of 5×10^{-7} with cosine decay, and a warmup over 10% of total steps. Training was run for 3 epochs with a batch size of one and gradient accumulation steps of 8. We used `bfloat16` precision and optimized training efficiency with DeepSpeed ZeRO Stage 3. We evaluated the model every 100 steps, and recorded logs for every 10 steps to monitor convergence behavior. We conducted training on 4 A100 80GB GPUs and took approximately 5 GPU-hours per model. The total compute budget, including data preprocessing and reward-based filtering, was approximately 20 GPU-hours.

13 Additional Results on Automatic Evaluation

13.1 Analysis of Training Data in DPO

To assess how much preference data is required for effective reward-based fine-tuning, we conduct an ablation study varying the number of training pairs used in RewardDPO across posters, websites. As shown in Table 16, performance improves steadily as training data increases from 1.2k (10%) to 4k (30–35%), but plateaus beyond that. Notably, models trained with just 4k preference pairs get near-saturation in all domains, yielding 97–99% of the gains observed with the full 12k set. For structured domains like posters and websites, edit adherence (EA), semantic consistency (SC), and layout similarity (LS) scores reach above 4.3 by this point.

These suggest that RewardDPO requires relatively modest supervision to internalize layout-sensitive preferences. Efficient learning is likely due to the structured nature of the reward signals and the contrastive framing of the optimization objective. This finding supports the scalability of preference-based finetuning in layout editing tasks—even in low-resource or domain-adaptive settings.

Training Size (Pairs)	Posters (EA / SC / LS)	Websites (EA / SC / LS)
10% (1.2k)	4.12 / 4.00 / 4.08	4.08 / 3.92 / 4.01
30% (4k)	**4.32 / 4.25 / 4.30**	**4.26 / 4.18 / 4.22**
100% (12k)	4.35 / 4.32 / 4.35	4.30 / 4.25 / 4.30

Table 16: **Ablation on RewardDPO training data volume across domains.** *Performance on both Posters and Websites saturates around 30% (around 4k pairs), with only marginal improvements beyond that scale.* Metrics: EA = Edit Adherence, SC = Semantic Consistency, LS = Layout Similarity.

Beam Width	Posters (EA / SC / LS)	Websites (EA / SC / LS)
1 (Greedy)	4.10 / 3.95 / 4.05	4.02 / 3.88 / 3.95
2	4.20 / 4.10 / 4.18	4.15 / 4.00 / 4.08
4	**4.32 / 4.25 / 4.30**	**4.26 / 4.18 / 4.22**
6	4.33 / 4.26 / 4.31	4.28 / 4.19 / 4.23
8	4.33 / 4.25 / 4.30	4.28 / 4.18 / 4.23

Table 17: **Beam search ablation across domains.** *Performance improves notably from greedy decoding (beam width 1) up to width 4, with diminishing returns beyond that point.* Metrics: EA = Edit Adherence, SC = Semantic Consistency, LS = Layout Similarity.

13.2 Beam Search Ablation

We ablate the beam width on layout edit quality across structured and unstructured domains. As shown in Table 17, increasing the beam size from 1 (greedy) to 4 leads to consistent improvements across all metrics, including Edit Adherence (EA), Semantic Consistency (SC), and Layout Similarity (LS). For example, in posters, performance improves from 4.05 (LS) under greedy decoding to 4.30 at beam size 4.

Beyond a beam width of 4, gains plateau across

Reward Configuration	Posters (SC / LS)	Websites (SC / LS)
All Rewards (Full)	**4.25 / 4.38**	**4.18 / 4.30**
w/o Alignment	4.10 / 4.22	4.05 / 4.16
w/o Semantic Order	4.05 / 4.18	3.98 / 4.10
w/o Cross-Sect. Consistency	4.08 / 4.15	4.00 / 4.12
w/o Overlap	4.22 / 4.28	4.15 / 4.25
w/o Whitespace	4.18 / 4.24	4.12 / 4.20
Only Semantic (SemOrder + XSec)	4.00 / 4.10	3.95 / 4.05
Only Spatial (Align + Overlap)	4.20 / 4.35	4.14 / 4.28

Table 18: **Ablation of reward components used in RefineLoop.** Performance is evaluated via GPT-4o against gold references. Semantic Consistency (SC) and Layout Similarity (LS) are rated on a 1–5 scale. *Removing alignment or semantic order yields the largest drops in structured domains, while both spatial and semantic feedback together yield the best overall gains.*

domains, indicating early convergence. Beam sizes larger than 6 offer minimal additional benefit while increasing computational cost. These suggest that beam width 4 offers a strong trade-off between performance and efficiency in layout-aware decoding.

13.3 Analysis of Reward Components in RefineLoop

To assess which reward signals are most critical for aligning model edits with gold references, we ablate individual components within RefineLoop and evaluate the resulting outputs using GPT-4o-based reference comparison. As shown in Table 18, removing *alignment* or *semantic order* results in the largest quality drops in structured domains, reflecting their importance in enforcing visual structure and narrative flow.

14 Validating GPT-4o Against Humans

To assess whether reliance on GPT-4o introduces systematic bias, we explicitly compared its outputs against human annotations rather than treating it as ground truth for two cases: **A) Poster Layout Parsing:** Two authors independently annotated 30 posters from SMARTEditBench into logical layout regions. Human–human agreement provides an empirical upper bound (IoU ≈ 0.82, F1 ≈ 0.88). GPT-4o's parsing aligns closely with the human consensus (IoU ≈ 0.80, F1 ≈ 0.87), falling well within normal human variability, **B) Reference-Based Evaluation as Judge.** GPT-4o is used as one of the reference-based judges for Semantic Consistency (SC) and Layout Similarity (LS). To validate its reliability, two authors independently evaluated 20 poster edits and 20 webpage edits using the same rubric and scale provided to GPT-4o. Agreement between GPT-4o and human evaluators (Pearson $r = 0.81$ for SC, 0.75 for LS) is only slightly below human–human agreement ($r = 0.88$ and 0.84), again placing GPT-4o within the natural range of human disagreement. These show that GPT-4o's judgments approximate human scoring fidelity at a level suitable for benchmarking.

15 Comments on Generalization

15.1 Model-Agnostic Optimization and Replaceability

Crucially, the learning signal in SMART-Editor is not derived from GPT-4o's generations but from *structured reward definitions*. In RewardDPO, preference pairs are constructed using (i) gold human edits when available and (ii) deterministic geometric degradations that introduce controlled violations (e.g., overlap, misalignment, broken narrative order). Negative examples are thus rule-driven rather than model-sampled.

As a result, once RewardDPO training is complete, the resulting policy executes in a single forward pass without invoking any critique model. GPT-4o therefore functions as a bootstrap mechanism during development and evaluation—not as a permanent dependency of the system.

15.2 Layout vs. Content Editing: Design Choice and Evaluation Focus

SMART-Editor is explicitly designed to support *both* layout and content editing. However, our evaluation emphasizes layout and structural coherence because this dimension is underexplored in prior work, while text quality has been extensively studied in language-centric benchmarks.

In real design workflows, content rewriting is typically localized, whereas layout edits introduce cascading global effects. Accordingly, SMARTEditBench prioritizes metrics such as narrative coherence, cross-sectional consistency, overlap, whitespace, and alignment—dimensions where existing models often fail even when textual edits are locally correct.

15.3 Content Robustness Beyond Layout

To verify that SMART-Editor improves semantic content quality rather than merely rearranging layout, we conducted a targeted robustness evaluation. We introduced controlled semantic corruptions into posters, including irrelevant sentence insertion, contradiction injection, and masking of key explanatory text. Figure 6 shows that visual–spatial

Algorithm 1 Reward-Guided Layout and Content Refinement

Input: Layout data D (e.g., webpage, PDF, or image), instruction I
Output: Final layout L^* and content C^*
Step 1: Layout Representation
Convert D into unified layout $L_0 = \{(o_i, \mathbf{b}_i, t_i)\}_{i=1}^{N}$ and initial content C_0
Step 2: Instruction Classification
Classify instruction I as `layout-only`, `semantic-only`, or `joint-edit`
Step 3: First-Pass Execution
Generate symbolic action plan $A = [a_1, \ldots, a_k]$ using an LLM
Apply actions to obtain layout L' and updated content C':

$$L', C' \leftarrow a_k(\ldots a_1(L_0, C_0))$$

Step 4: Reward Evaluation
Compute layout reward: $R(L') = \sum_k \lambda_k \cdot r_k(L')$
if $R(L') < \tau$ or constraints are violated **then**
    Initialize $L^{(0)} \leftarrow L'$, $C^{(0)} \leftarrow C'$
    for $t = 1$ to T **do**
        Generate k layout-content candidates via beam search on $(L^{(t-1)}, C^{(t-1)}, I)$
        Select best candidate: $(L^{(t)}, C^{(t)}) \leftarrow \arg\max R(L)$
        if $L^{(t)}$ satisfies all constraints **then**
            return $L^* = L^{(t)}, C^* = C^{(t)}$
        end if
        Generate feedback f_t (e.g., "Results overlaps with Discussion")
        Refine via LLM: $(L^{(t)}, C^{(t)}) \leftarrow$ `LLM_Refine`$(L^{(t)}, C^{(t)}, I, f_t)$
    end for
end if
return $L^* = L'$ or $L^{(T)}$, $C^* = C'$ or $C^{(T)}$

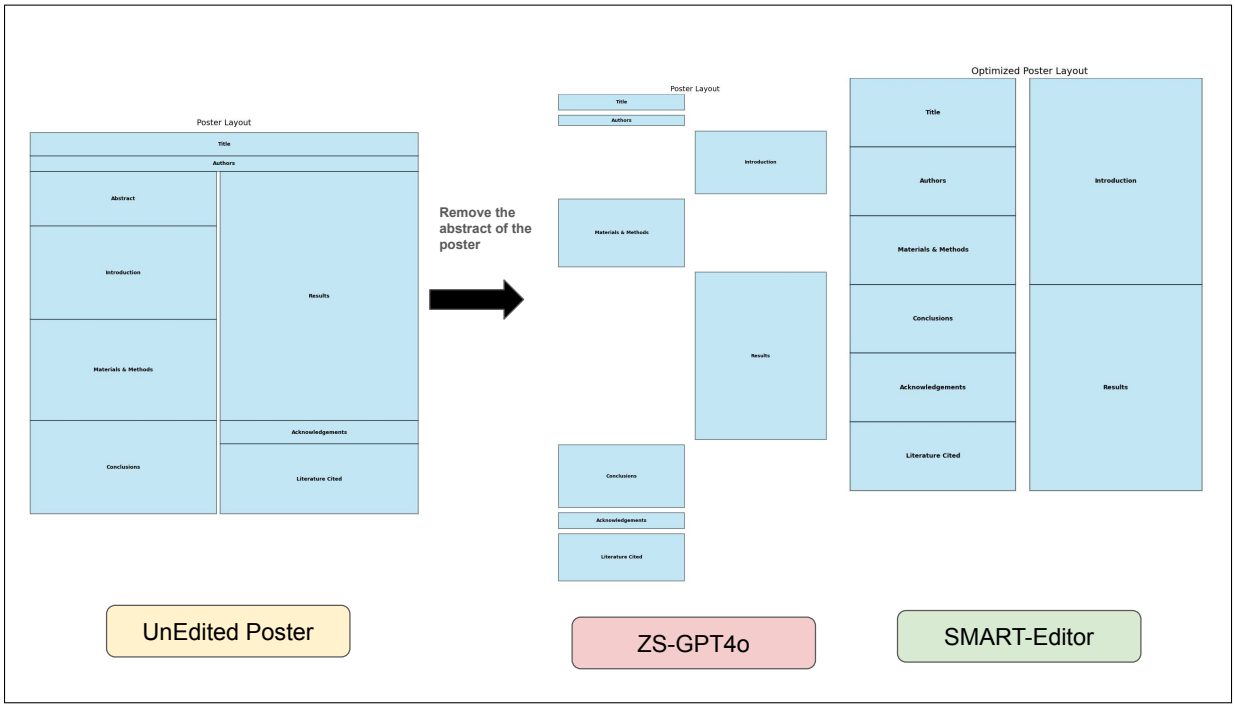

Figure 7: GPT-4o (ZS baseline) had higher penalties: Overlap penalty of 0, Wspc penalty of 0.35, and alignment reward only about 0.55. SMART-Editor clearly improved: Ovlp same as 0, Wspc tightened to 0.12, and alignment reward increased to 0.78.

rewards increase sharply across refinement iterations, while semantic rewards improve more modestly.

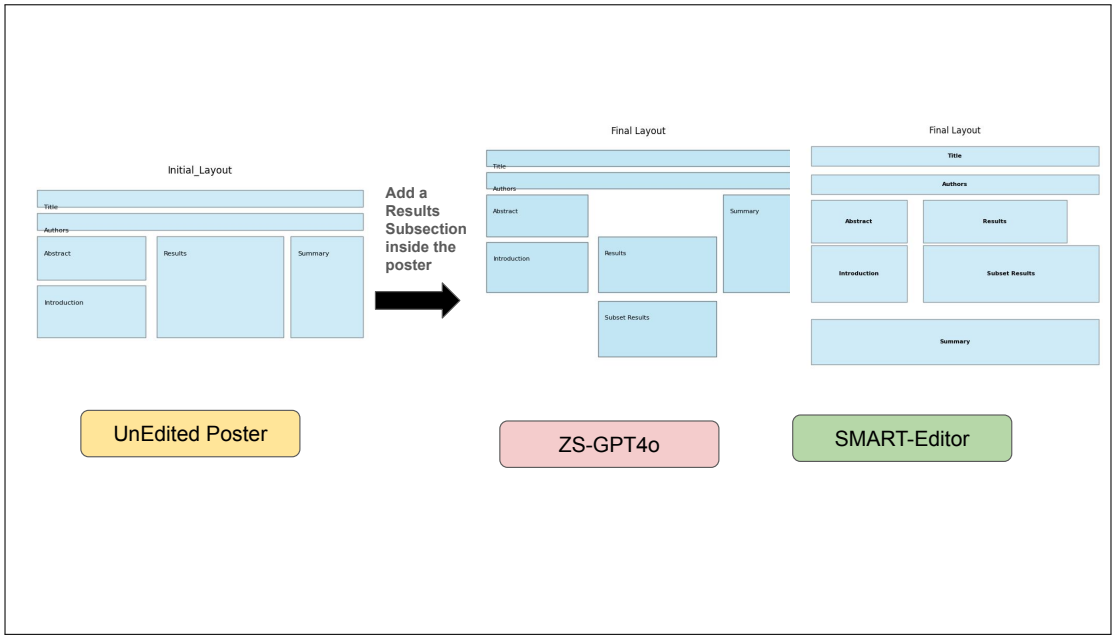

Figure 8: GPT-4o (ZS baseline) shows higher penalties on Whitespace, and alignment reward of 0.52. However, adding the Results subsection after the Conclusion breaks the natural top-down reading order, so narrative coherence drops. SMART-Editor improves whitespace while boosting alignment reward and narrative coherence by putting the summary section below.

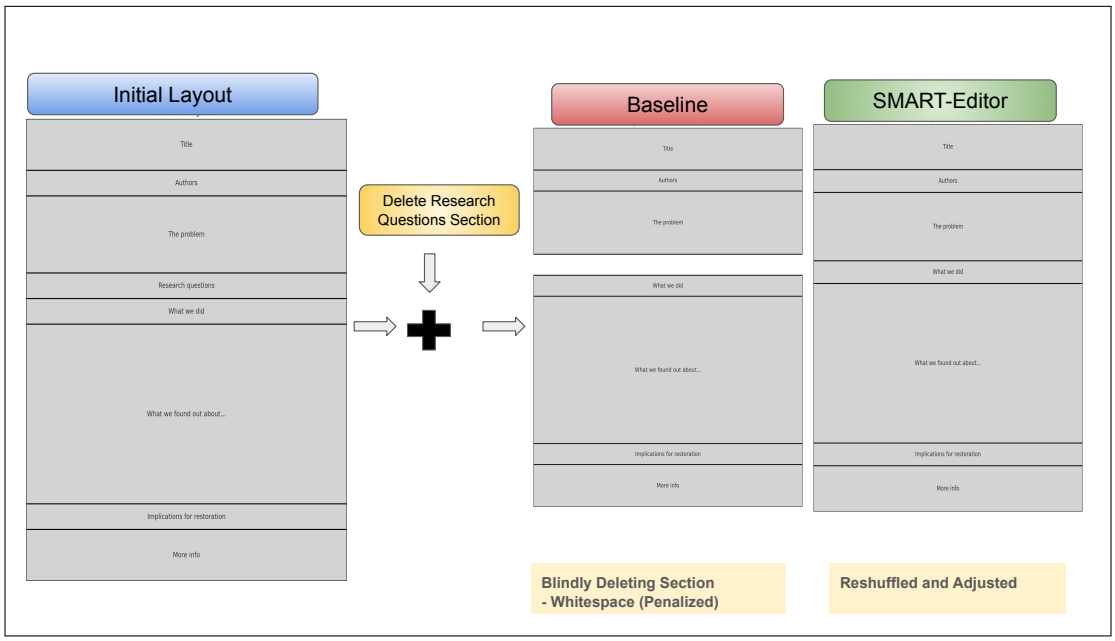

Figure 9: GPT-4o handles the deletion of the Research Questions section poorly, leaving excessive whitespace and disrupting balance, which penalizes compactness and readability. SMART-Editor not only deletes the section but also reshuffles and realigns the remaining content, reducing wasted space and improving visual structure

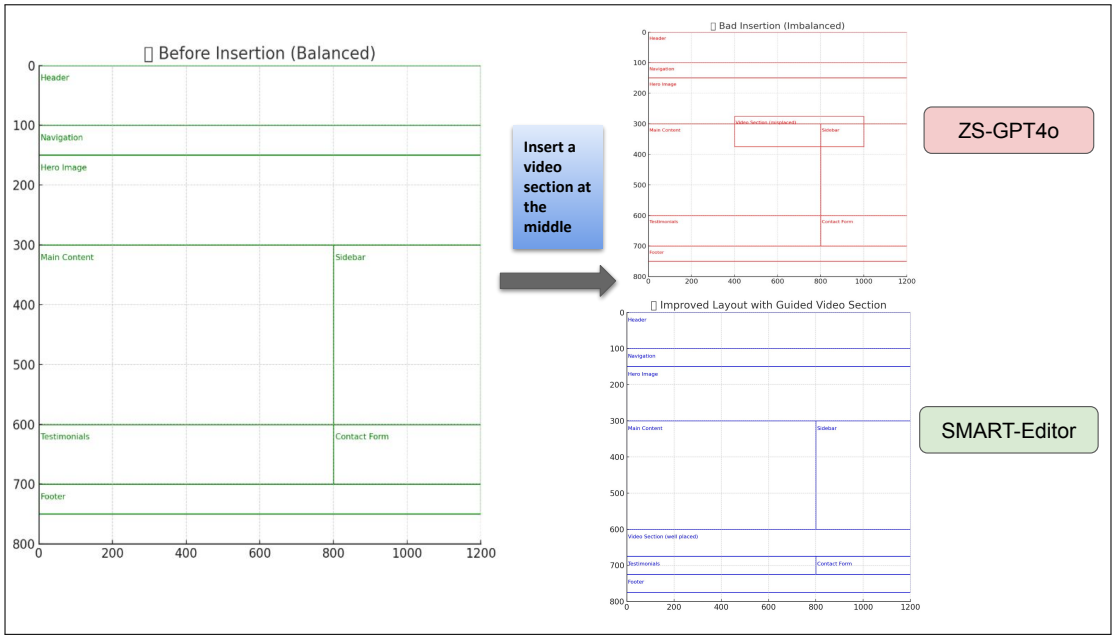

Figure 10: Baseline output (ZS-GPT4o): The video section is dropped into the canvas without much adjustment. This causes nearby sections to collide and leaves uneven whitespace around it, making the layout harder to scan and less visually organized. Alignment across sections is inconsistent, so the poster feels fragmented. SMART-Editor: The video section is inserted more thoughtfully — spacing is redistributed, overlaps are minimized, and sections align better along the grid. This improves readability and visual balance across the poster. The initial Figure is clearly shown in Figure 11 and the edited figures are clearly shown in Figure 12.

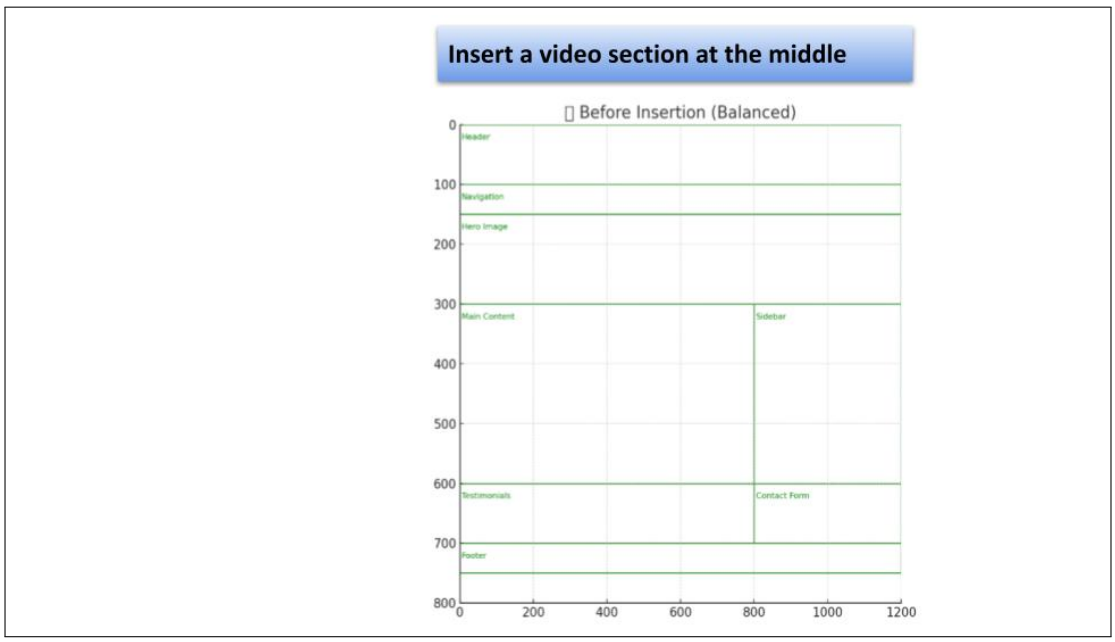

Figure 11: Part-a (Unedited Picture)

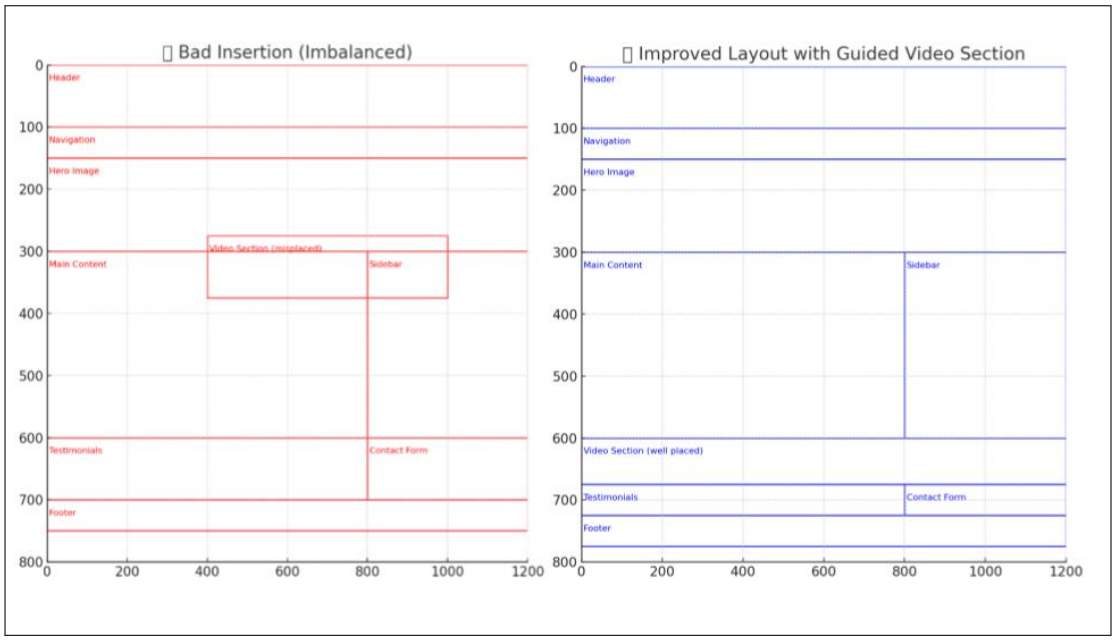

Figure 12: Part-b (Edited Pictures).

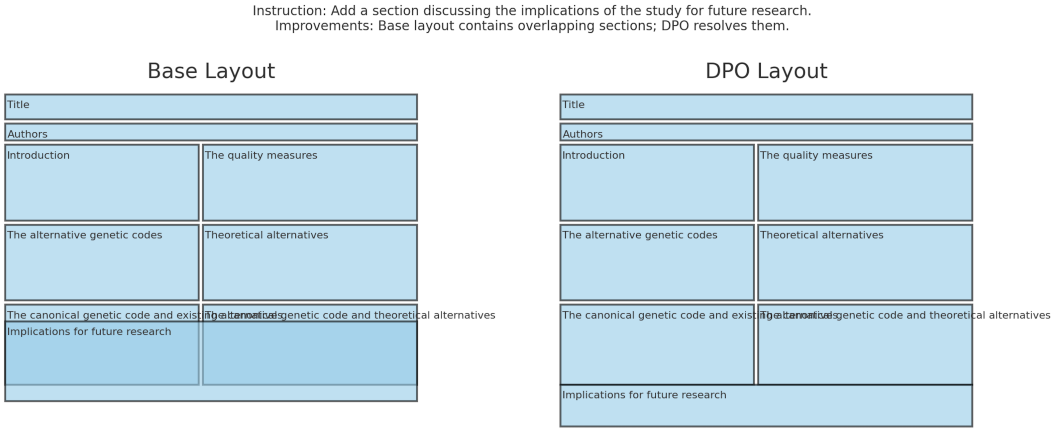

Figure 13: Poster edit example: RewardDPO fixes overlap and restores alignment after inserting a new section.

Algorithm 2 Object Translation and Insertion

function TRANSLATEOBJECT($layout, section\_id, dx, dy$)
  $[x_1, y_1, x_2, y_2] \leftarrow layout[section\_id].bbox$
  $layout[section\_id].bbox \leftarrow [x_1 + dx, y_1 + dy, x_2 + dx, y_2 + dy]$
  return $layout$
end function
function SHIFTOBJECTSAFTER($layout, anchor\_id, dy$)
  $anchor\_y \leftarrow layout[anchor\_id].bbox[1]$
  for all $(object\_id, object)$ in $layout$ **do**
    if $object.bbox[1] \geq anchor\_y$ **then**
      $layout \leftarrow$ TRANSLATEOBJECT($layout, object\_id, 0, dy$)
    end if
  end for
  return $layout$
end function
function INSERTOBJECT($layout, object\_id, bbox, text$)
  $layout[object\_id] \leftarrow \{\texttt{bbox} : bbox, \texttt{text} : text\}$
  return $layout$
end function

Algorithm 3 Text Update, Bounding Box, and Reordering

function UPDATEOBJECTTEXT($layout, object\_id, new\_text$)
  $layout[object\_id].text \leftarrow new\_text$
  return $layout$
end function
function REPLACEOBJECTBBOX($layout, object\_id, new\_bbox$)
  $layout[object\_id].bbox \leftarrow new\_bbox$
  return $layout$
end function
function REORDEROBJECTS($layout, order, x\_start, width, y\_padding$)
  $y \leftarrow 100$
  for all $object\_id$ in $order$ **do**
    $height \leftarrow layout[object\_id].bbox[3] - layout[object\_id].bbox[1]$
    $layout[object\_id].bbox \leftarrow [x\_start, y, x\_start + width, y + height]$
    $y \leftarrow y + height + y\_padding$
  end for
  return $layout$
end function

Algorithm 4 object Removal, Resizing, and Alignment

function REMOVEOBJECT($layout, object\_id$)
  delete $layout[object\_id]$
  return $layout$
end function
function RESIZEOBJECT($layout, object\_id, scale\_x, scale\_y$)
  $[x_1, y_1, x_2, y_2] \leftarrow layout[object\_id].bbox$
  $cx \leftarrow (x_1 + x_2)/2$, $cy \leftarrow (y_1 + y_2)/2$
  $width \leftarrow (x_2 - x_1) \cdot scale\_x$
  $height \leftarrow (y_2 - y_1) \cdot scale\_y$
  $layout[object\_id].bbox \leftarrow [cx - width/2, cy - height/2, cx + width/2, cy + height/2]$
  return $layout$
end function
function ALIGNOBJECTLEFT($layout, object\_id, x\_ref$)
  $[x_1, y_1, x_2, y_2] \leftarrow layout[object\_id].bbox$
  $width \leftarrow x_2 - x_1$
  $layout[object\_id].bbox \leftarrow [x\_ref, y_1, x\_ref + width, y_2]$
  return $layout$
end function

Algorithm 5 Centering and Snapping to Grid

function CENTEROBJECT($layout, object\_id, canvas\_width$)
  $[x_1, y_1, x_2, y_2] \leftarrow layout[object\_id].bbox$
  $width \leftarrow x_2 - x_1$
  $cx \leftarrow canvas\_width/2$
  $layout[object\_id].bbox \leftarrow [cx - width/2, y_1, cx + width/2, y_2]$
  return $layout$
end function
function SNAPTOGRID($layout, object\_id, grid\_unit$)
  $bbox \leftarrow layout[object\_id].bbox$
  $snapped \leftarrow$ round each coordinate in $bbox$ to nearest multiple of $grid\_unit$
  $layout[object\_id].bbox \leftarrow snapped$
  return $layout$
end function