RAP: A DATASET FOR PAPER-TO-POSTER GENERATION VIA DECLARATIVE MARKUP FROM HUMAN-CRAFTED EXEMPLARS

Anonymous authors
Paper under double-blind review

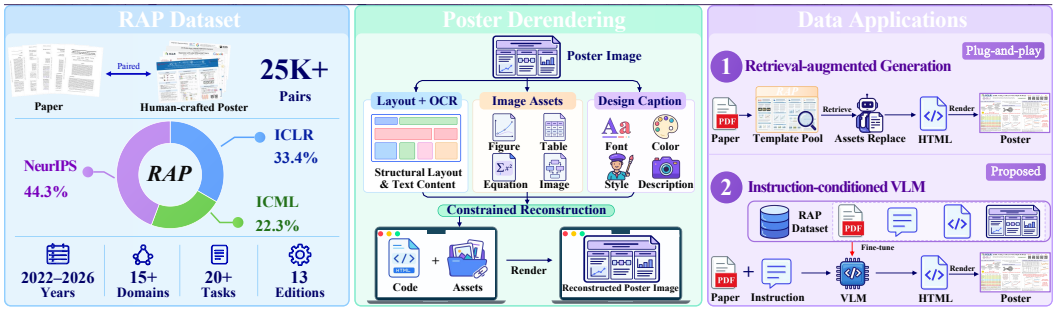

Figure 1: Overview of *RAP*. **Left:** paired scientific papers and human-crafted posters from major machine learning conferences. **Middle:** automated derendering reconstructs posters as editable HTML with linked assets. **Right:** the resulting data support retrieval-augmented template adaptation and instruction-conditioned VLM post-training for HTML-based poster generation.

ABSTRACT

Making a conference poster is visual coding in miniature: an entire paper must be compressed onto a single page, which exercises summarization, visual design, and code implementation at once, using nothing beyond the paper itself. Yet benchmarks for this task in the context of visual coding remain scarce: existing resources are small and annotated only with layout boxes, so evaluation must fall back on VLM-as-judge or human checklists rather than objective render-and-compare, and the generation methods they support design layouts from scratch, leaving human-crafted posters unusable as either a retrievable design library or as verified supervision. We introduce RAP (Reconstruct A Poster), a dataset and benchmark for paper-to-poster generation as a visual coding task. *RAP* contains 25K+ verified paper-poster pairs from various Machine Learning conferences published between 2022 and 2026, annotated with a three-level content taxonomy, plus a held-out benchmark of 150 test papers each paired with its human poster. A fully automated derendering pipeline turns each poster into styled HTML/CSS of typed, refillable slots, so every sample is a paper-poster-markup triplet verifiable by rendering against the source poster. The corpus serves both inference and training: PosterRAG, a training-free multi-agent pipeline, retrieves an exemplar and refills its slots without touching layout or styling, while an SFT+RL recipe trains a dedicated model from the same triplets. An online demo is available at `https://reconstruct-a-poster.github.io/`.

1 INTRODUCTION

Automatically condensing a scientific paper into a conference poster is an ideal yet long-overlooked validation task for visual coding. The task exercises three capabilities that are individually measurable—summarization, layout and visual design, and code implementation: a model must extract the salient content of a full-length paper, organize that content into a clear layout with coherent

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Comparison of representative scientific poster datasets and benchmarks. Our resource links full papers and human-crafted poster exemplars to reconstructed, styled HTML with typed, refillable content slots. *P2PInstruct counts instruction–response pairs with synthetic instruction/HTML targets; human-crafted posters serve as generation references. ‡Full-paper coverage is partial: 100 posters in SciPostLayout; SciPostLayoutTree inherits partial coverage through SciPostLayout.

Dataset	Scale	Full Paper	Renderable Markup	Human-crafted Poster	Annotations
SciPostLayout (Tanaka et al., 2024)	7,855	✓‡	✗	✓	Boxes
PosterSum (Saxena et al., 2025)	16,305	✗	✗	✓	Abstracts, topic labels
Paper2Poster (Pang et al., 2026)	100	✓	✗	✓	QA pairs
P2PEval (Sun et al., 2026)	121	✓	✗	✓	Checklists
SciPostLayoutTree (Tanaka et al., 2026)	7,849	✓‡	✗	✓	Boxes, OCR, reading order, hierarchy
SciPostGen (Inadumi et al., 2026)	18,097	✓	✗	✓	Boxes, OCR, figure/table links
P2PInstruct (Sun et al., 2026)	30,460	✗	✓	✗*	Instructions, figure descriptions, text, HTML
***RAP* (Ours)**	**25,435**	✓	✓	✓	**Boxes, OCR, HTML, figure assets, asset links, design captions, taxonomy**

visual design under limited space and a fixed aspect ratio, and render the result legibly for human readers. All three draw solely on information that the paper itself supplies, so every model operates on identical inputs, free from the variance introduced by web search.

However, existing benchmarks remain limited in scale, annotation depth, and utilization. **First, scale and verifiability.** Paper2Poster provides 100 paper-poster pairs (Pang et al., 2026) and P2PEval 121 (Sun et al., 2026). Larger collections supply abstracts (Saxena et al., 2025) or full papers for only 100 of 7,855 posters (Tanaka et al., 2024). SciPostGen pairs 18,097 papers and posters but targets layout-level annotation (Inadumi et al., 2026). Generation benchmarks rely on subjective, costly VLM judgments or human checklists (Pang et al., 2026; Sun et al., 2026) rather than direct render-and-compare verification, limiting reproducibility and statistical robustness. **Second, task formulation and annotation depth.** Resources annotate boxes, categories, reading order, and hierarchy (Tanaka et al., 2024; 2026), but lack renderable markup reconstructing source posters for end-to-end visual coding. **Third, utilization.** Methods largely generate layouts from scratch, while a template-based system retrieves from a few curated slides (Zheng et al., 2025). P2PInstruct supplies model-generated rather than human-grounded HTML designs (Sun et al., 2026). Existing resources lack reusable human-crafted design libraries and verified paper-poster supervision for generator training. Table 1 summarizes these gaps.

We therefore introduce **RAP (Reconstruct A Poster)**, a dataset and benchmark for paper-to-poster generation as a visual coding task. At its core is the largest paper-poster collection to date: 25,435 verified pairs collected from NeurIPS, ICML, and ICLR (2022–2026) via OpenReview, each enriched with paper-level metadata, a distilled introduction summary, and annotations from a three-level taxonomy of 15 domains, 20 tasks, and 39 methods that support content-aware retrieval and analysis. Complementing the corpus, we curate a held-out benchmark of 150 test papers, 10 per domain, each paired with its ground-truth human poster. Because every sample carries its own human-crafted reference, render-and-compare verification is available at the scale of the whole corpus rather than only on a handful of curated examples, making measurement objective and reproducible instead of dependent on judge-based scoring.

To demonstrate the dataset's utility at both inference time and training time, we develop two applications that exploit its complementary supervision signals. *(1) Training-free retrieval-augmented generation.* We present **PosterRAG**, a multi-agent pipeline that treats the dataset as a reusable design library. A hierarchical retrieval cascade—categorical filtering, narrative matching, and aesthetic scoring—selects the best-matching exemplar for an input paper; a mapping agent then programmatically refills the retrieved template's typed slots with paper content while leaving all layout and styling untouched; and a VLM reviewer iteratively repairs residual visual defects. Because PosterRAG inherits human-crafted designs unmodified rather than hallucinating layouts, it achieves state-of-the-art diversity and distributional human-likeness against strong baselines. *(2) Post-training a dedicated paper-to-poster model.* The triplets likewise provide direct input–output supervision: we apply supervised fine-tuning (SFT) to a multimodal model on paper–markup pairs, then further optimize it with reinforcement learning using verifiable rewards (Shao et al., 2024).

Our contributions are as follows:

- **A large-scale dataset and benchmark.** *RAP* is the largest paper-poster resource to date, comprising 25,435 verified pairs with three-level content annotations and a held-out benchmark of 150 test papers. Because every sample carries a human-crafted reference, evaluation proceeds by objective render-and-compare rather than by judge-based scoring alone.
- **An automated derendering pipeline.** A fully automated pipeline converts raster posters into styled, renderable HTML/CSS in which every element is an addressable, typed, and refillable slot.
- **Two dataset-driven applications.** PosterRAG, a training-free retrieval-augmented pipeline, and an SFT+RL recipe for training a dedicated paper-to-poster model demonstrate that the same corpus serves as a reusable design library at inference time and as verified supervision at training time.

2 Related Work

Visual Coding Data and Benchmarks. Visual coding has emerged as a rigorous testbed for multimodal models. pix2code (Beltramelli, 2018) pioneers code generation from GUI screenshots; Design2Code (Si et al., 2025) and Vision2Web (He et al., 2026) extend the task to real webpages and multi-page websites, the latter incorporating agent-based verification; Plot2Code (Wu et al., 2025) and ChartMimic (Yang et al., 2025b) target scientific plots and charts. These benchmarks, however, all overlook the paper-to-poster task: their inputs are screenshots or short specifications of a single, self-contained view, whereas poster generation requires condensing a full-length paper into a space-constrained, information-dense design and implementing it as code. *RAP* is constructed around the requirements of visual coding: the full paper supplies the long-context input, every poster is derendered into styled, renderable HTML/CSS with typed, refillable slots, and reconstruction fidelity is measured by direct render-and-compare against the human reference, making paper-to-poster, for the first time, a visual coding benchmark whose outputs are verifiable at scale.

Paper-to-Poster Generation. Existing resources for this task supply posters and, at best, layout-level annotations. Paper2Poster (Pang et al., 2026) and P2PEval (Sun et al., 2026) pair papers with only a few hundred raw poster images and rely on VLM judges for evaluation; PosterSum (Saxena et al., 2025) pairs posters with abstracts rather than full papers; SciPostLayout and SciPostLayout-Tree (Tanaka et al., 2024; 2026) annotate layout boxes, reading order, and figure correspondence, and SciPostGen (Inadumi et al., 2026) does so at larger scale, but none provides renderable markup that reconstructs the poster. Approaches to generation fall into two families. *Text-to-image generation.* Recent progress in visual text rendering (Chen et al., 2024; Tuo et al., 2024; Peng et al., 2025) and in frontier image generators (OpenAI, 2025; Google, 2025; Gao et al., 2025; Black Forest Labs, 2024; Ideogram AI, 2024) enables the synthesis of poster-like images directly from prompts. Their outputs, however, are non-editable raster images whose text fidelity degrades sharply on dense scientific content, and they offer no guarantee that the rendered content faithfully reflects the source paper. *LLM-based pipelines.* Paper2Poster (Pang et al., 2026), P2P (Sun et al., 2026), and PosterGen (Zhang et al., 2026) decompose the task into paper parsing, content planning, and markup rendering in HTML or PPTX, typically with multi-agent checking and refinement, whereas PosterLLaVa (Yang et al., 2024) and PosterLlama (Seol et al., 2024) fine-tune multimodal models for the content-aware layout sub-problem in isolation. Instruction-tuning data for the task is model-generated rather than human-grounded: P2PInstruct (Sun et al., 2026) supplies HTML targets synthesized by models rather than derived from human-crafted posters. Document-to-slide systems (Fu et al., 2022; Zheng et al., 2025; Jin et al., 2026) follow the same parse–plan–render paradigm; in particular, PPTAgent (Zheng et al., 2025) retrieves from a small set of manually curated slide templates, but its rigid filling mechanism often fails to adapt content appropriately. Both families thus confront a common trade-off: either sacrifice editability and content fidelity, or generate layouts from scratch without grounding in human design. *RAP* addresses both sides of this trade-off: its derendered, human-crafted posters ground a retrieval-augmented pipeline that inherits professional designs rather than inventing them, and its paper-poster-markup triplets provide direct supervision for post-training a dedicated generation model with reference-based rewards.

3 *RAP*: Large-Scale Poster Derendering

Data Collection and Statistics. RAP dataset is built from accepted papers at three top-tier ML conferences from 2022 through 2026. Using OpenReview, we collect raw poster files together with

their associated paper records. For each submission, we retrieve three types of information: *(i)* the paper PDF or LaTeX source when available from supplementary material; *(ii)* the author-uploaded poster file in either PDF or image format; and *(iii)* metadata including title, authors, abstract, venue, year, and keywords. We then normalize poster formats and match each poster to its source paper, yielding a final dataset of 25,435 verified paper–poster pairs distributed across NeurIPS (11,256), ICLR (8,504), and ICML (5,675). Each paper is further enriched with structured annotations, including a taxonomy of 15 domains, 20 tasks, and 39 methods, plus a distilled introduction summary for narrative-level retrieval (§B.1.1).

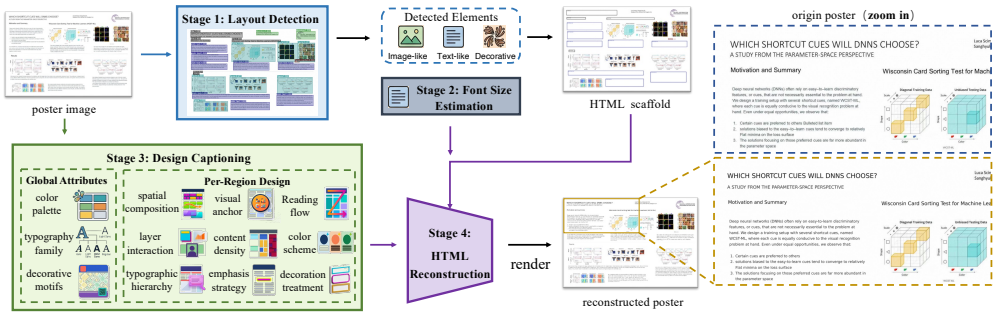

Figure 2: Derendering pipeline illustration.

Derendering Pipeline. Our derendering pipeline converts a raw poster image into a structured HTML template with typed, fillable slots through four stages (Figure 2): **Stage 1: Layout Detection.** A fine-tuned LayoutLMv3 (Huang et al., 2022) model localizes all visual elements (Title, Plain Text, Figure, Table, Formula, Caption, etc.) with bounding boxes and confidence scores. Each detected element is subsequently assigned to one of three semantic categories: `text-like` (titles, plain texts, captions), `image-like` (figures, tables, formulae) or `decorative` (icons, logos). **Stage 2: Font Size Estimation.** For each `text-like` bounding box, the corresponding image region is cropped and processed by PaddleOCR (Du et al., 2020). Text lines are identified by grouping OCR detection results via y-axis overlap, and the font size in pixels is estimated as $\text{font_size} = (h_{\text{bbox}}/n_{\text{lines}}) \times 0.8$, where the factor 0.8 accounts for inter-line spacing, yielding a conservative estimate directly usable as CSS `font-size`. **Stage 3: Design Captioning.** the original poster image alone is passed to a MLLM, which produces a structured design caption encoding both global attributes (the page-level color palette, typography family, and decorative motifs) and per-region design dimensions: spatial composition, visual anchor, reading flow, layer interaction, content density, color-scheme roles, typographic hierarchy, emphasis strategy, and border/decoration treatment. **Stage 4: HTML Reconstruction.** An HTML scaffold is first programmatically constructed with absolute-positioned `<div>` elements for each detected block: `image-like` and `decorative` regions are cropped from the original image and embedded as `<img>` elements, while `text-like` blocks are rendered as empty content slots with their OCR-measured font sizes pre-annotated as styling constraints. The scaffold encodes pixel-exact block geometry but leaves all visual styling (colors, backgrounds, typography, and decorative CSS) as empty placeholders to be filled. The scaffold, the original poster image, and the design caption are then jointly provided to a MLLM, which populates each `text-like` slot by reading content directly from the poster, applies the pre-annotated font sizes and per-region descriptors, and synthesizes CSS decorations to reproduce the poster's visual appearance.

The derendering outcome is a complete HTML file in which each content block is a fully styled, addressable slot that faithfully reproduces the original poster's layout and visual appearance.

4 METHOD

Building on our RAP dataset, we develop two applications for high-quality poster generation: *PosterRAG: Training-free retrieval-augmented generation*, which retrieves and adapts structured templates from our poster bank, and *Post-training a dedicated paper-to-poster model* with our dataset.

4.1 POSTERRAG: TRAINING-FREE RETRIEVAL-AUGMENTED GENERATION

As illustrated in Figure 3, PosterRAG consists of two stages: *Template Retrieval* and *Poster Generation*. Given an input paper P in LaTeX or PDF format, Stage 1 constructs its structured representation

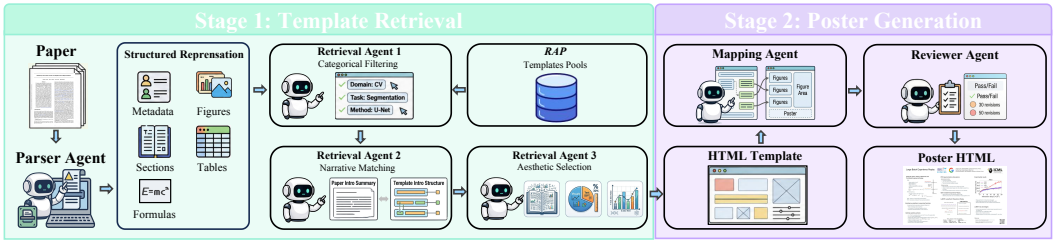

Figure 3: Two-stage PosterRAG pipeline for multi-agent template retrieval and poster generation.

P_{meta} with a **ParserAgent** and selects the most suitable template $T^\star$ from the template bank $\mathcal{T}$ through a three-level **RetrievalAgent** cascade. Given $T^\star$ and P_{meta}, Stage 2 produces the final poster $\hat{T}$ using a **MappingAgent** followed by an iterative **ReviewerAgent**.

ParserAgent. Following Paper2Poster (Pang et al., 2026), the ParserAgent takes the input paper as input and converts it into a structured representation $P_{\text{meta}} = \{P_d, P_t, P_m, P_s, P_{\text{parse}}\}$, where P_d, P_t, and P_m denote the domain, task, and method labels of the paper, respectively; P_s the narrative summary, and P_{parse} the parsed paper content, including text, figures, tables, and formulas.

RetrievalAgent-1: Categorical filtering. RetrievalAgent-1 retains templates whose domain, task, and method labels match (P_d, P_t, P_m). Papers with identical labels often share structural patterns, making their templates compatible with the input paper.

RetrievalAgent-2: Narrative Matching. RetrievalAgent-2 re-ranks the filtered candidates by comparing the input paper's introduction summary P_s with the pre-computed summary T_s^i of each template's source paper. This comparison captures how papers frame their contributions and emphasize content. For example, a paper following a *gap-and-fill* narrative is better matched with a template organized around the same problem–solution structure than with one based on an *observation-driven* narrative. This favors templates whose organization and section emphasis align with the input paper.

RetrievalAgent-3: Aesthetic Selection. RetrievalAgent-3 selects the final template by scoring each candidate poster image I_{T^i} on three independent 1–5 scales: ***Content Density***, ranging from *key-point* to *full*; ***Visualization Ratio***, ranging from *text-heavy* to *figure-heavy*; and ***Scientific Aesthetics***, ranging from *bare-bones* to *custom-designed*. These scores form $\mathbf{s}_{T^i} \in \{1, \ldots, 5\}^3$. Without user preferences, the candidate closest to the neutral midpoint is selected; otherwise, a VLM converts the preferences into dimension weights $\mathbf{w}$ to guide the selection.

MappingAgent. The MappingAgent fills the typed slots $\{s_1, \ldots, s_m\}$ of the retrieved template $T^\star$ with content from P_{meta} in three phases. ***Slot analysis*** records each slot's ID, type, label, position, and dimensions, estimating text capacity from the slot size and font size and recording the available area for images. ***Content mapping and adaptation*** provides an LLM with this slot summary, the importance-ranked paper sections, figures and tables with captions, and pre-rendered assets $\mathcal{R}$. The LLM produces $\mathcal{M} = \{(s_j, \text{act}_j, \text{content}_j)\}$, where each action replaces, retains, or clears a slot, and the mapped content is adapted to its capacity and section type. ***Programmatic filling*** then applies $\mathcal{M}$ through a deterministic dispatcher: text and image slots receive new `innerHTML` and `src` values, while unused slots are hidden. During filling, the original layout, typography, colors, and decorative styling remain unchanged, preserving the human-crafted design and full editability.

ReviewerAgent. Following Paper2Poster (Pang et al., 2026), the ReviewerAgent takes the rendered HTML poster $\hat{T}$ as input and performs iterative visual refinement through a *see-then-fix* process.

4.2 Post-training a dedicated paper-to-poster model

As illustrated in Fig. 4, we construct training signals from the paper and its reference poster, including paper information and the corresponding user instruction, which serve as input to post-training. The resulting VLM enables users to generate posters tailored to their preferred design and layout.

A. Paper Information Extraction. We construct the paper-side input from two complementary components: **Paper Text** and **Paper Assets**. The paper text is extracted from method, experiments, and conclusion sections, with section-specific budgets used to retain content most relevant to poster generation. For paper assets, we detect tables, images, figures and equations from the PDF, assign

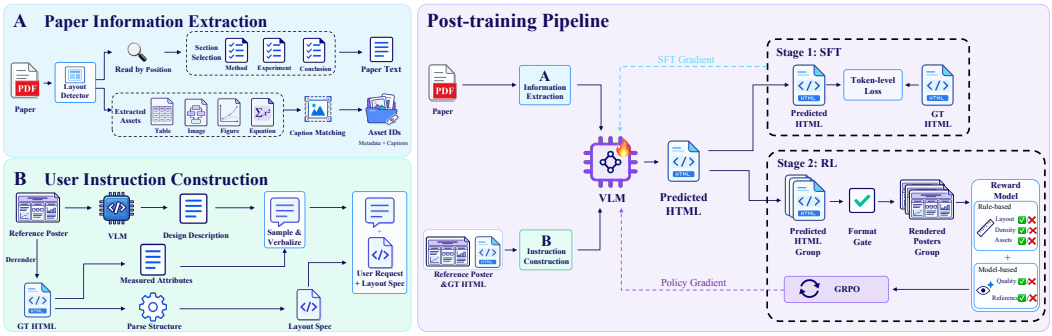

Figure 4: Post-Training pipeline for Vision-Language Model.

unique identifiers, and associate captions based on spatial proximity. We then match the assets to visual slots in the reference poster through visual feature matching and geometric verification, using VLM adjudication for ambiguous cases and leaving uncertain matches unresolved. Assets used in the target poster are retained with their identifiers, metadata, captions, and thumbnails under a fixed visual-token budget. The resulting paper-side input is

$$X_{\text{paper}} = (T_{\text{paper}}, \mathcal{A}_{\text{paper}}), \tag{1}$$

where T_{paper} denotes the section-organized paper text and $\mathcal{A}_{\text{paper}}$ denotes the selected assets and their associated information.

B. User Instruction Construction. We construct the instruction-side input from two complementary components: **Design** and **Layout**. For design, the reference poster is processed by a VLM to obtain a design description covering its color palette, typography, composition, and visual emphasis. In parallel, attributes such as canvas size and visual density are measured from the ground-truth HTML. The design description and measured attributes are then sampled and verbalized with varying levels of detail and writing styles to form a natural-language user request. For layout, the ground-truth HTML is parsed to recover the columns, section order, block geometry, text–image composition, and asset assignments. This structure is converted into a layout specification without exposing the exact poster text or HTML implementation. The resulting instruction-side input is

$$X_{\text{instruction}} = (u_{\text{design}}, s_{\text{layout}}), \tag{2}$$

where u_{design} is the natural-language user request and s_{layout} is the structured layout specification.

Post-training Pipeline. We first perform supervised fine-tuning, followed by rendering-aware reinforcement learning with GRPO. During RL, multiple HTML candidates $\{H_i\}_{i=1}^{G}$ are sampled and rendered into poster images $I_i = \mathcal{R}(H_i)$, each receiving a composite reward:

$$R_i = g(H_i)\left[\lambda_r R_{\text{rule}}(H_i) + \lambda_v R_{\text{vlm}}(I_i, I^\star)\right], \tag{3}$$

where $g(H_i) \in \{0, 1\}$ is a validity gate that rejects invalid or non-renderable HTML, $I^\star$ is the reference poster, and $\lambda_r = 0.4$ and $\lambda_v = 0.6$ weight the rule-based and model-based rewards, respectively. **Rule-based reward** rule uses deterministic measurements from the HTML and its rendered Document Object Model (DOM), which provides each element's position, size, and style. It contains three groups. ***Layout*** checks text clipping, overlap, out-of-canvas elements, and block correspondence with the reference layout. ***Density*** evaluates block and text coverage, empty regions, body font size, and the number of text blocks, preventing sparse outputs produced by deleting content or shrinking text. ***Assets*** checks identifier validity, required-figure coverage, broken images, and excess empty image slots. Together, these checks enforce valid layout, sufficient content, and correct asset usage. **Model-based reward** vlm uses a VLM to evaluate the rendered candidate given the reference poster, paper information, and user instruction. It contains two groups. ***Quality*** evaluates rendering integrity, legibility, layout, density, content completeness and faithfulness, figure usage, visual design, and instruction adherence. ***Reference*** evaluates structural similarity in columns, section order, and block placement, as well as stylistic similarity in colors, typography, and decorations. The weighted dimension scores are reduced by capped penalties for visible defects, providing perceptual assessments of overall quality and reference fidelity.

Table 2: Quantitative results for the two dataset-driven poster generation approaches. Best results are in **bold**, and second-best results are underlined.

(a) Template Retrieval-based Generation

Method	Diversity			Human-likeness	
	LPIPS↑	DINOv2-CosDist↑	DISTS↑	FID↓	FDiD↓
NanoBanana Pro Google (2025)	0.5695	0.2364	0.2231	24.25	483.97
P2P Sun et al. (2026)	0.5761	0.2847	0.2340	33.24	377.25
Paper2Poster Pang et al. (2026)	0.5547	0.1138	0.2107	81.59	1156.73
PosterGen Zhang et al. (2026)	0.4131	0.2710	0.2527	133.83	682.27
Ours (Image Guidance)	0.5584	0.0877	0.2246	77.11	1267.84
Ours (HTML Guidance)	**0.6471**	**0.4443**	**0.3490**	**23.75**	**306.79**

(b) Post-trained Paper-to-Poster Models

Model	Diversity			Human-likeness	
	LPIPS↑	DINOv2-CosDist↑	DISTS↑	FID↓	FDiD↓
GPT-6-Astra OpenAI (2026)	0.5704	0.2093	0.2369	27.64	341.64
Kimi-K3 Kimi Team (2026)	0.5856	0.1974	0.2445	27.94	340.50
Gemini-3.1-Pro Google DeepMind (2026)	0.5956	0.1983	0.2365	41.49	408.05
Claude-Opus-5 Anthropic (2026b)	0.6414	0.2320	0.2728	**21.56**	**289.49**
Qwen3.8-27B Qwen Team (2026)	0.6284	0.2579	0.2680	32.44	465.00
Qwen3.8-27B-SFT (Ours)	0.6599	**0.2784**	0.2892	28.00	326.24
Qwen3.8-27B-RL (Ours)	**0.6925**	0.2735	**0.2944**	23.42	306.94

5 EXPERIMENTS

5.1 BASELINES AND DATASET

We compare against Nano Banana, P2P (Sun et al., 2026), Paper2Poster (Pang et al., 2026), and PosterGen (Zhang et al., 2026). To ensure a fair comparison, all LLM-based agent calls across all methods are replaced with Claude-Opus4.6 (Anthropic, 2026a). Our evaluation uses a held-out test set of papers not included in the 25K template bank. Specifically, we construct the test set of 150 samples by collecting 10 papers from each of the 15 research domains in *RAP*-25K.

5.2 EVALUATION METRICS

To evaluate the diversity and human-likeness of generated posters, we assess along two axes.

Diversity. We measure cross-paper generation diversity by computing pairwise LPIPS (Zhang et al., 2018), DINOv2-CosDist, and DISTS (Ding et al., 2020) over all pairs of generated posters in the test set, then averaging across pairs. LPIPS captures perceptual diversity via AlexNet features; DINOv2-CosDist measures semantic-level diversity using cosine distances between DINOv2 (Oquab et al., 2023) embeddings; DISTS captures deep structural and textural diversity via VGG features. Higher values indicate more varied generations.

Human-likeness. We measure the distributional gap between generated posters and our *RAP*-25K collection using Fréchet Inception Distance (**FID**) (Heusel et al., 2017) and Fréchet DINOv2 Distance (**FDiD**). FDiD replaces InceptionV3 (Szegedy et al., 2016) with DINOv2 (Oquab et al., 2023) features, making it sensitive to visual structure and layout geometry. Together, these two metrics quantify the distribution-level fidelity of generated posters to real human-crafted designs.

5.3 MAIN RESULTS

5.3.1 QUANTITATIVE COMPARISON

Table 2(a) shows that RAP (HTML Guidance) achieves the best scores across all five metrics. It attains FID/FDiD of 23.75/306.79, with FDiD 18.7% lower than P2P (377.25), the best baseline on this metric. Its DINOv2-CosDist reaches 0.4443, compared with 0.2847 for P2P and 0.1138 for Paper2Poster. These results support retrieval and adaptation of human-crafted templates for improving both cross-paper diversity and distributional human-likeness.

Table 2(b) shows that RAP-based SFT improves all five metrics over the base Qwen3.8-27B. Subsequent RL reduces FID/FDiD from 28.00/326.24 to 23.42/306.94 and increases LPIPS and DISTS, with a small decrease in DINOv2-CosDist. Among the compared models, our RL variant achieves the highest LPIPS and DISTS and the second-lowest FID/FDiD, behind Claude-Opus-5. These results demonstrate the utility of RAP for improving cross-paper diversity and distributional alignment with human-crafted posters through post-training.

5.3.2 QUALITATIVE COMPARISON

Figure 5 compares posters generated from the same paper. Paper2Poster uses a conventional column-based layout, whereas NanoBanana Pro produces a more stylized raster poster. RAP (HTML Guidance) follows the retrieved template's color palette, typographic hierarchy, and section orga-

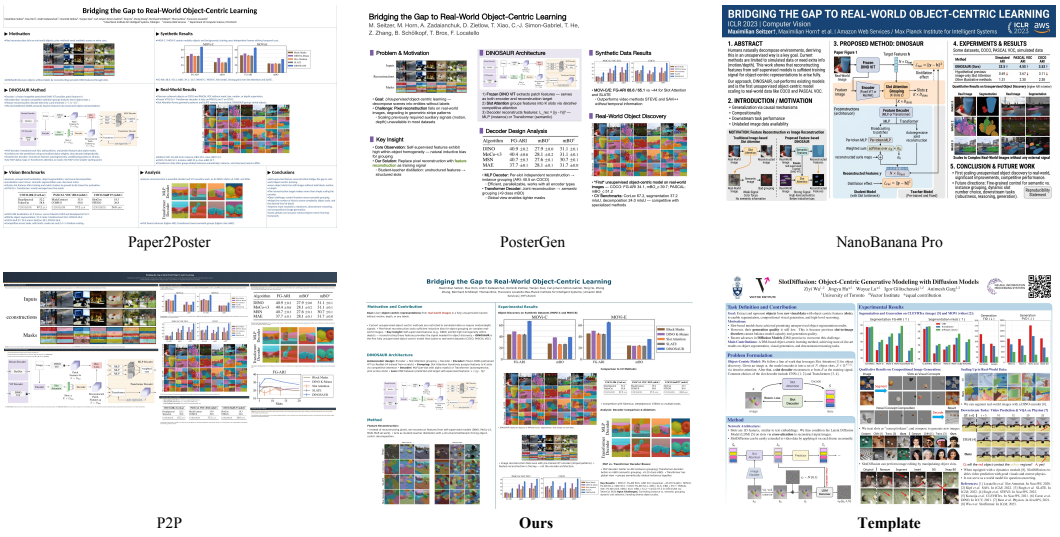

Figure 5: Qualitative comparison between PosterRAG (ours) and baseline methods.

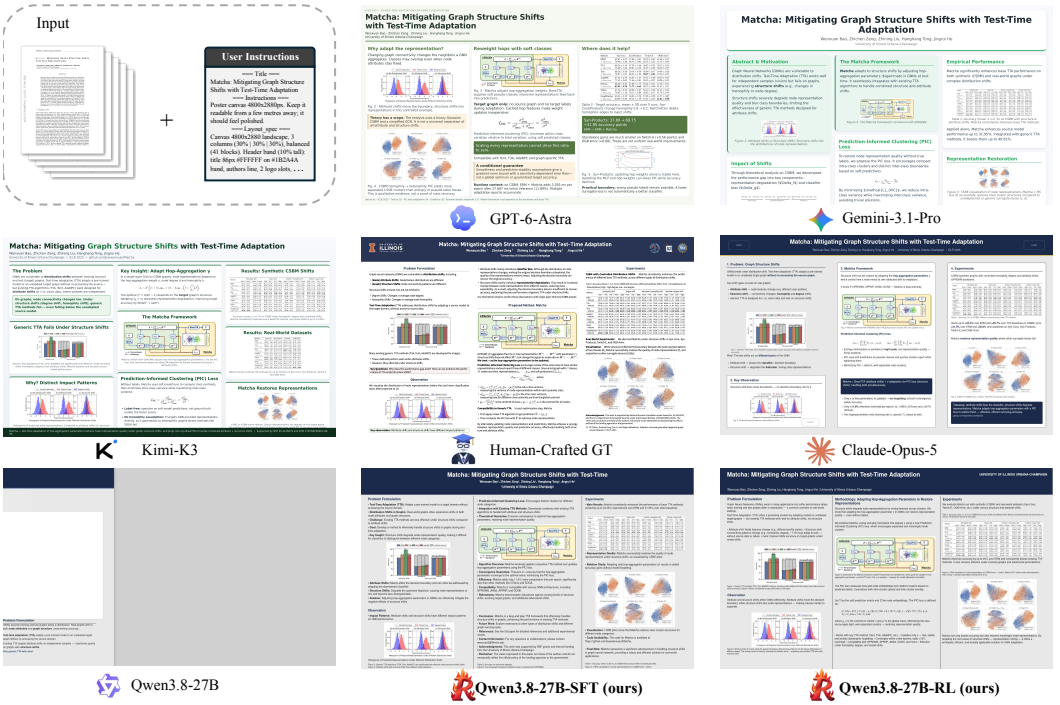

Figure 6: Qualitative comparison between our post-trained model and other vision-language models.

nization while adapting its content to the input paper. This example illustrates how human-crafted templates provide reusable design guidance for paper-to-poster generation.

Figure 6 compares outputs for the same paper and user instruction. The base Qwen3.8-27B produces a largely incomplete poster, whereas both post-trained variants organize text, figures, and tables into complete three-column layouts. Their dark title bands and column organization follow the supplied design specification and resemble the human-crafted reference. This example illustrates improved layout adherence and more complete rendered outputs after post-training.

5.4 Ablation Studies

RL reward-component ablations are reported in Appendix B.6.

Ablation on HTML Generation. We compare *Ours (HTML Guidance)*, which fills retrieved HTML templates, with *Ours (Image Guidance)*, which generates HTML from poster-image references. HTML Guidance improves all five metrics, reducing FID/FDiD from 77.11/1267.84 to 23.75/306.79. DINOv2-CosDist rises from 0.0877 to 0.4443, alongside higher LPIPS (0.5584 to 0.6471) and DISTS (0.2246 to 0.3490). Unlike image-only guidance, the HTML scaffold explicitly constrains layout and styling while allowing content replacement and refinement. These results support using reconstructed HTML, rather than the reference image alone, to improve distributional human-likeness and cross-paper diversity.

Ablation on derendering. For this poster reconstruction task, we adopt five metrics: **SSIM** Wang et al. (2004) measures pixel-level structural similarity. **DISTS** Ding et al. (2020) measures deep structural and texture dissimilarity. **Layout mIoU** computes mean IoU between detected and ground-truth layout elements, in both class-aware (mIoU) and class-agnostic (mIoU_{ag}) variants. **FID** (Heusel et al., 2017) measures the distributional fidelity to the original poster set.

Table 3: Ablation studies on the derendering pipeline. Best results in **bold**.

(a) Backbone Model

	Pairwise				Fidelity	
	SSIM↑	DISTS↓	mIoU↑	mIoU_{ag}↑	FID↓	FDiD↓
GPT-4o	0.7928	0.1643	0.4692	0.5081	41.00	371.41
Qwen-3.5	0.7922	0.0966	0.4349	0.4702	28.86	120.18
Gemini-3.1-Pro	**0.8076**	**0.0623**	0.6400	0.6706	17.30	66.28
Claude-Opus-4.6	0.7896	0.0663	**0.6897**	**0.7246**	**16.90**	**61.71**

(b) Pipeline Components (backbone: Claude-Opus-4.6)

	Pairwise				Fidelity	
	SSIM↑	DISTS↓	mIoU↑	mIoU_{ag}↑	FID↓	FDiD↓
Base	0.5843	0.4265	0.0652	0.1069	53.63	977.20
+Layout	0.7744	0.0857	0.5236	0.5590	18.32	93.48
+Layout +OCR	0.7859	0.0686	0.6808	0.7148	17.42	65.49
+Layout +OCR +Design (Ours)	**0.7896**	**0.0663**	**0.6897**	**0.7246**	**16.90**	**61.71**

Backbone model. Table 3(a) compares four backbone MLLMs under the full pipeline. Claude-Opus-4.6 (Anthropic, 2026a) and Gemini-3.1-Pro (Google DeepMind, 2026) clearly outperform GPT-4o (Hurst et al., 2024) and Qwen-3.5 (Yang et al., 2025a). Gemini-3.1-Pro achieves the best pixel-level fidelity (SSIM, DISTS), while Claude-Opus-4.6 leads on layout accuracy and distributional fidelity, suggesting it better preserves the structural organization of the original poster. We therefore adopt Claude-Opus-4.6 as the backbone for all other experiments.

Pipeline components. Table 3(b) shows that adding Layout yields the largest improvement across all metrics, raising mIoU from 0.0652 to 0.5236. OCR further improves layout accuracy and distributional fidelity, while Design Caption provides smaller but consistent gains across all six metrics. The full pipeline achieves the best reconstruction scores, supporting the complementary roles of geometric, typographic, and design constraints.

6 Conclusion

We have cast paper-to-poster generation as a visual coding problem and constructed the resource it requires. RAP (Reconstruct A Poster) contains 25,435 verified paper-poster pairs from NeurIPS, ICML, and ICLR (2022–2026), annotated along a three-level taxonomy of 15 domains, 20 tasks, and 39 methods, together with a held-out benchmark of 150 papers whose human posters are withheld for evaluation. A fully automated derendering pipeline rewrites each poster as styled HTML/CSS with typed, refillable slots, reconstructing it faithfully enough (SSIM 0.79, layout mIoU 0.69) that fidelity can be measured by rendering and comparing rather than by judge scoring.

These paper-poster-markup triplets support two applications. PosterRAG is training-free: it retrieves a human-designed template and refills its slots without disturbing the layout or styling, and it leads every baseline on both diversity and human-likeness. An SFT+RL recipe trains a dedicated model against the same references. Together they show that human designs are not merely evaluation yardsticks but a reusable source of design knowledge. We release the dataset, derendering pipeline, codebase, and an online demo to make reference-grounded visual coding a reproducible setting for future research.

AI USE STATEMENT

Claude-Opus-4.6 powered our derendering and RAG pipelines and the agentic baselines; derendering ablations additionally evaluated GPT-4o, Qwen-3.5, and Gemini-3.1-Pro. NanoBanana Pro served as an image-generation baseline. We post-trained Qwen3.8-27B through SFT and RL, using Gemini-3.8-Flash for VLM-based rewards, and compared against its base model, GPT-6-Astra, Kimi-K3, Gemini-3.1-Pro, and Claude-Opus-5. VLMs also assisted training-data construction. AI assistants, including Codex, supported manuscript drafting and language polishing. We retain full responsibility for all research decisions and submitted content.

REFERENCES

Anthropic. Claude opus 4.6 system card. https://www.anthropic.com/claude-opus-4-6-system-card, 2026a.

Anthropic. Claude opus 5 system card. https://www.anthropic.com/claude-opus-5-system-card, 2026b.

Tony Beltramelli. pix2code: Generating code from a graphical user interface screenshot. In *Proceedings of the ACM SIGCHI symposium on engineering interactive computing systems*, pp. 1–6, 2018.

Black Forest Labs. Flux.1: A 12 billion parameter text-to-image generation model, 2024. https://blackforestlabs.ai.

Jingye Chen, Yupan Huang, Tengchao Lv, Lei Cui, Qifeng Chen, and Furu Wei. Textdiffuser-2: Unleashing the power of language models for text rendering. In *European Conference on Computer Vision*, pp. 386–402. Springer, 2024.

Keyan Ding, Kede Ma, Shiqi Wang, and Eero P Simoncelli. Image quality assessment: Unifying structure and texture similarity. *IEEE transactions on pattern analysis and machine intelligence*, 44(5):2567–2581, 2020.

Yuning Du, Chenxia Li, Ruoyu Guo, Xiaoting Yin, Weiwei Liu, Jun Zhou, Yifan Bai, Zilin Yu, Yehua Yang, Qingqing Dang, et al. Pp-ocr: A practical ultra lightweight ocr system. *arXiv preprint arXiv:2009.09941*, 2020.

Tsu-Jui Fu, William Yang Wang, Daniel McDuff, and Yale Song. Doc2ppt: Automatic presentation slides generation from scientific documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 634–642, 2022.

Yu Gao, Lixue Gong, Qiushan Guo, Xiaoxia Hou, Zhichao Lai, Fanshi Li, Liang Li, Xiaochen Lian, Chao Liao, Liyang Liu, et al. Seedream 3.0 technical report. *arXiv preprint arXiv:2504.11346*, 2025.

Google. Nano banana, 2025. https://ai.google.dev/gemini-api/docs/image-generation?hl=zh-cn.

Google DeepMind. Gemini 3.1 pro model card. https://deepmind.google/models/model-cards/gemini-3-1-pro, 2026.

Zehai He, Wenyi Hong, Zhen Yang, Ziyang Pan, Mingdao Liu, Xiaotao Gu, and Jie Tang. Vision2web: A hierarchical benchmark for visual website development with agent verification. *arXiv preprint arXiv:2603.26648*, 2026.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM international conference on multimedia*, pp. 4083–4091, 2022.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Ideogram AI. Ideogram 2.0: Text-to-image generation with superior text rendering, 2024. https://ideogram.ai.

Shun Inadumi, Shohei Tanaka, Tosho Hirasawa, Atsushi Hashimoto, Koichiro Yoshino, and Yoshitaka Ushiku. Scipostgen: Bridging the gap between scientific papers and poster layouts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2131–2141, 2026.

Yiqiao Jin, Rachneet Kaur, Zhen Zeng, Sumitra Ganesh, and Srijan Kumar. Slideagent: Hierarchical agentic framework for multi-page visual document understanding. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14858–14881, 2026.

Kimi Team. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

OpenAI. Gpt-image, 2025. gpt-image-1 (April 2025) and gpt-image-2 (April 2026). https://platform.openai.com/docs/models.

OpenAI. Gpt-6 astra system card. https://deploymentsafety.openai.com/gpt-6-astra/, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Wei Pang, Kevin Qinghong Lin, Xiangru Jian, Xi He, and Philip Torr. Paper2poster: Towards multimodal poster automation from scientific papers. *Advances in Neural Information Processing Systems*, 38, 2026.

Yuyang Peng, Shishi Xiao, Keming Wu, Qisheng Liao, Bohan Chen, Kevin Lin, Danqing Huang, Ji Li, and Yuhui Yuan. Bizgen: Advancing article-level visual text rendering for infographics generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23615–23624. IEEE, 2025.

Qwen Team. Qwen3.8-27b. https://huggingface.co/Qwen/Qwen3.8-27B, 2026.

Rohit Saxena, Pasquale Minervini, and Frank Keller. Postersum: A multimodal benchmark for scientific poster summarization. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 1828–1844, 2025.

Jaejung Seol, Seojun Kim, and Jaejun Yoo. Posterllama: Bridging design ability of language model to content-aware layout generation. In *European Conference on Computer Vision*, pp. 451–468. Springer, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. 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)*, pp. 3956–3974, 2025.

Tao Sun, Enhao Pan, Zhengkai Yang, Kaixin Sui, Jiajun Shi, Xianfu Cheng, Tongliang Li, Ge Zhang, Wenhao Huang, Jian Yang, et al. P2p: Automated paper-to-poster generation and fine-grained benchmark. In *International Conference on Learning Representations*, volume 2026, pp. 18756–18798, 2026.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

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

Shohei Tanaka, Atsushi Hashimoto, and Yoshitaka Ushiku. Scipostlayouttree: A dataset for structural analysis of scientific posters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2753–2762, 2026.

Yuxiang Tuo, Wangmeng Xiang, Jun-Yan He, Yifeng Geng, and Xuansong Xie. Anytext: Multilingual visual text generation and editing. In *International Conference on Learning Representations*, volume 2024, pp. 56783–56799, 2024.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Chengyue Wu, Zhixuan Liang, Yixiao Ge, Qiushan Guo, Zeyu Lu, Jiahao Wang, Ying Shan, and Ping Luo. Plot2code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3006–3028, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran Xu, Xinyu Zhu, Siheng Li, Yuxiang Zhang, et al. Chartmimic: Evaluating lmm's cross-modal reasoning capability via chart-to-code generation. In *International Conference on Learning Representations*, volume 2025, pp. 26590–26646, 2025b.

Tao Yang, Yingmin Luo, Zhongang Qi, Yang Wu, Ying Shan, and Chang Wen Chen. Posterllava: Constructing a unified multi-modal layout generator with llm. *arXiv preprint arXiv:2406.02884*, 2024.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Zhilin Zhang, Xiang Zhang, Jiaqi Wei, Yiwei Xu, and Chenyu You. Postergen: Aesthetic-aware multi-modal paper-to-poster generation via multi-agent llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9813–9823, 2026.

Hao Zheng, Xinyan Guan, Hao Kong, Wenkai Zhang, Jia Zheng, Weixiang Zhou, Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. Pptagent: Generating and evaluating presentations beyond text-to-slides. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14402–14418, 2025.

A Additional Dataset Statistics

A.1 Dataset Overview and Taxonomy Distribution

RAP-25K annotates each paper with a three-level taxonomy comprising 15 domains, 20 tasks, and 39 methods. Table 4 lists all domain categories with their counts and proportions.

Alongside the domain statistics, Figure 7 provides a visual overview of human-crafted posters in RAP. Their varied layouts and visual styles form the reference pool used by our retrieval and post-training applications.

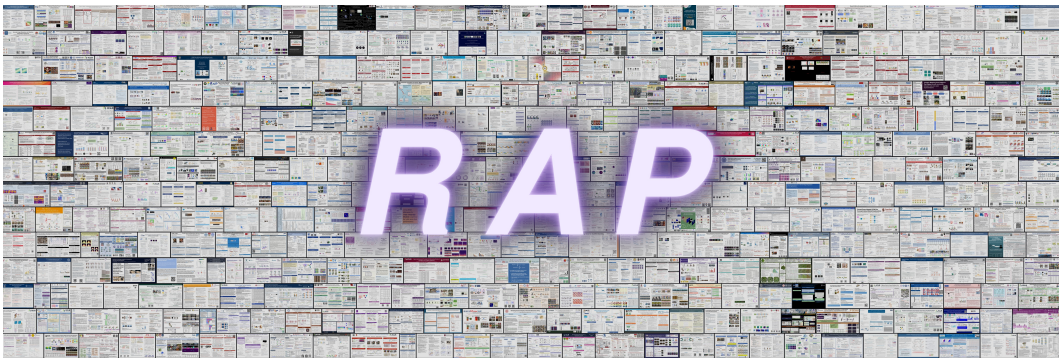

Figure 7: Overview of human-crafted posters in RAP. The mosaic illustrates variation in layout, color palette, typography, and content density.

Table 4: Full domain taxonomy of *RAP*-25K (15 categories).

Domain	**Count**	**%**
Optimization & Learning Theory	3,487	13.7
AI Safety & Alignment	2,580	10.1
Computer Vision	2,414	9.5
Reinforcement Learning	2,312	9.1
Multimodal & Cross-modal Learning	2,164	8.5
NLP & LLMs	1,966	7.7
Representation & Transfer Learning	1,870	7.4
AI for Natural Sciences	1,672	6.6
Probabilistic & Generative ML	1,665	6.6
Graph & Relational Learning	992	3.9
Data-centric ML & Evaluation	926	3.6
ML Systems & Infrastructure	899	3.5
AI for Health & Social Sciences	894	3.5
Robotics & Embodied AI	872	3.4
Fairness & Social Impact	722	2.8
Total	**25,435**	**100.0**

A.2 Per-Conference Breakdown

Table 5 provides the detailed per-conference and per-year breakdown for *RAP*-25K. Note that for 2026, only ICLR data is currently included.

Table 5: Per-conference and per-year poster counts.

Conference	2022	2023	2024	2025	2026	Total
NeurIPS	2,280	2,613	3,037	3,326	0	11,256
ICLR	831	889	1,434	2,085	3,265	8,504
ICML	882	1,471	1,751	1,571	0	5,675
Total	**3,993**	**4,973**	**6,222**	**6,982**	**3,265**	**25,435**

B Implementation Details

B.1 Method

B.1.1 PosterRAG: Training-free retrieval-augmented generation

Stage 1: Template Retrieval. Stage 1 selects the most suitable poster template T^* from the template bank $\mathcal{T}$ for a given input paper.

ParserAgent. The ParserAgent converts the raw paper into a structured representation P_{meta} that drives all downstream retrieval and generation. The parser supports both LaTeX source and PDF input, extracting metadata, segmenting content by section boundaries, and building a figure registry $\mathcal{F}$ of all image assets in PNG format for HTML embedding; the result is a parsed content object $P_{\text{parse}} = \{\text{meta}, \text{abstract}, \text{sections}, \text{figures}, \text{tables}, \text{formulas}\}$. Beyond content extraction, the ParserAgent classifies the paper into a three-level taxonomy by prompting an LLM with the abstract and section titles, producing a *domain* label P_d, a *task* label P_t, and a *method* label P_m. It also distills the introduction into a structured summary P_{s} along four dimensions: motivation, problem formulation, key insight, and claimed contributions, which serves as the semantic fingerprint for narrative-level template matching. Together these form $P_{\text{meta}} = \{P_d,\ P_t,\ P_m,\ P_s,\ P_{\text{parse}}\}$.

RetrievalAgent-1: Categorical filtering. RetrievalAgent-1 performs a hard filter, retaining only those templates whose categorical labels exactly match the input paper's domain, task, and method. The rationale is that papers sharing the same domain, task, and method tend to follow similar paper structures and argue their contributions in comparable ways, making their posters naturally compatible in layout and content organization.

RetrievalAgent-2: Narrative Matching. Beyond category-level structural compatibility, this stage performs finer-grained selection by comparing how papers frame their contributions and what content they emphasize. Each template's source paper is associated with a pre-computed introduction summary T_s^i extracted during dataset construction; RetrievalAgent-2 measures the narrative similarity between the input paper's P_s and each candidate's T_s^i, and re-ranks the candidates accordingly. This captures whether two papers share similar narrative logic in their introductions. For instance, a paper following a *gap-and-fill* structure ("existing methods fail to handle X; we propose Y to address this") pairs better with a template whose source paper argues along the same problem–solution arc, rather than one structured as an *observation-driven* narrative ("experiments reveal that phenomenon X occurs under condition Y; motivated by this finding, we derive Z"). By matching at the narrative level, RetrievalAgent-2 ensures that the selected template's content layout and section emphasis align with how the input paper tells its story.

RetrievalAgent-3: Aesthetic selection. The final retrieval stage selects a single template by assessing visual style rather than content. RetrievalAgent-3 scores the poster image I_{T^i} of each candidate on three mutually independent dimensions on a 1–5 integer scale, producing a score vector $\mathbf{s}_{T^i} \in \{1, \ldots, 5\}^3$: **Content Density**, from *key-point* to *full*, capturing how much of the paper's content is reproduced; **Visualization Ratio**, from *text-heavy* to *figure-heavy*, capturing the balance between figures/tables and textual content; and **Scientific Aesthetics**, from *bare-bones* to *custom-designed*, capturing the level of decoration and visual polish. Without user preference, the candidate closest to the neutral midpoint is selected as a safe default. When style preferences are provided, an VLM-based intent recognizer converts them into per-dimension weights $\mathbf{w}$, shifting selection toward user-emphasized dimensions.

Stage 2: Poster Generation. Given the retrieved template T^* and the parsed paper content P_{meta}, Stage 2 produces the final poster via a **MappingAgent** that fills the template with paper content, followed by an iterative **ReviewerAgent** that refines visual quality.

MappingAgent. The MappingAgent takes the retrieved template T^* with its typed slots $\{s_1, s_2, \ldots, s_m\}$ and the parsed paper P_{meta}, and produces a filled poster through three consecutive phases. ***Slot analysis.*** The agent profiles each template slot programmatically. For text-like slots, it computes an approximate character capacity derived from their display dimensions (w_j, h_j) and font size f_j; for image-like slots, it records the available dimensions for figure sizing. The result is a slot summary describing each slot's ID, type, label, spatial position, dimensions, and text capacity. ***Content mapping and adaptation.*** An LLM receives the slot summary alongside the paper's section content with importance scores, figure and table list with captions, and any pre-rendered equation or table assets $\mathcal{R}$. It produces a mapping $\mathcal{M} = \{(s_j, \text{act}_j, \text{content}_j)\}$ for every slot, where $\text{act}_j \in \{\texttt{replace_text}, \texttt{replace_image}, \texttt{replace_rendered}, \texttt{keep}, \texttt{clear}\}$ is the fill action, and content_j is adapted from the corresponding source paper element to fit the slot's capacity and formatted according to section type (e.g., bullet points for methods, flowing prose for introductions). ***Programmatic filling.*** The core insight of our approach is that filling is performed *programmatically* via direct manipulation rather than LLM-based HTML regeneration. Using a deterministic action dispatcher, only slot-level content is updated: text slots receive new `innerHTML`, image slots receive new `src` attributes from the figure registry $\mathcal{F}$, and unmapped or stale slots are hidden via visibility flags. All CSS rules, spatial layout, typography, color schemes, gradients, and decorative elements remain exactly as in the original human-created template. This structure-preserving invariant ensures that RAP inherits professional visual design quality while maintaining full editability.

ReviewerAgent. Although programmatic filling preserves the template's global structure, inserting new content into fixed slots may still cause visual artifacts such as text overflow, font-size mismatches, poor color contrast, or awkward whitespace. The ReviewerAgent resolves these issues through an iterative *see-then-fix* loop where each round renders the current HTML poster $\hat{T}$ into a screenshot and uses VLM feedback to guide localized refinements. ***Visual review.*** At each round r, a headless Chromium browser renders $\hat{T}$ to a pixel-accurate screenshot I_r. The VLM receives I_r along with past fixes, returning an overall score score_r and a list of block-level repair suggestions $\{(b_j, \text{issue}_j, \text{patch}_j)\}$ specifying the target block ID, issue type, and a targeted CSS patch. ***Constrained refinement.*** Each patch_j is applied locally to its target block b_j in $\hat{T}$ under strict safety constraints that explicitly block destructive modifications across four categories: visibility-hiding rules that would render content invisible to users, text-hostile overflow and truncation properties that cause text to be clipped or hidden, layout geometry alterations that would shift or resize elements, and structural changes to the DOM or color scheme that exceed the intended scope of local refinement. ***Termination and design rationale.*** The loop terminates once $\text{score}_r \geq \tau_{\text{score}}$ (default: 8/10), $N_{\max} = 3$ rounds are reached, or no further patches are proposed by the agent. Two decisions distinguish this agent from the prior methods (Pang et al., 2026; Sun et al., 2026): *programmatic CSS patching* rather than full HTML regeneration ensures layout stability, and *screenshot-level feedback* detects rendering-dependent defects invisible from source code alone.

B.2 Prompt Templates

B.2.1 Prompt 1: Design Caption in Derendering

Used in Stage A of the derendering pipeline to extract a structured design intermediate representation (IR) from a poster image, *without* any pre-computed bounding boxes.

System prompt. *"You are a senior visual designer and information architect specialised in academic posters. Return a strict JSON description that downstream code can render back to HTML faithfully."*

User prompt. The MLLM receives a single input: the original poster image. It is asked to identify high-level semantic regions entirely from visual observation (no layout annotations are provided) and, for each region, assign design tokens drawn from a fixed vocabulary of **10 design dimensions** (9 per-region + 1 global):

- `global_decorations`: *solid_block_carrier / thin_divider_lines / logo_placeholder / qr_placeholder / background_pattern*
- `composition`: *grid_2col / grid_3col / flex_row / flex_col / absolute_overlap*
- `visual_anchor`: *scale_dominant / weight_dominant / contrast_dominant / isolated_focal*
- `flow`: *z_pattern / f_pattern / top_down / left_right*
- `interaction`: *solid_carrier / offset_underlay / partial_overlay / glass_morphism / flat_inline*
- `density`: *airy / balanced / dense*
- `color_scheme`: *neutral_palette / warm_palette / cool_palette*
- `typography`: *sans_modern / serif_classic / mono_technical*
- `emphasis`: *bold_caps / color_tint / icon_prefix / highlight_bg*
- `decoration`: *border_solid_square / border_dashed_square / border_solid_rounded / border_none_shadow / border_thick_accent*

Key rules.

- **Region granularity**: use **3–7 regions**, one per major section (Header / Motivation / Method / Results / Conclusion). Sub-figures, columns, and sub-panels within the same section must *not* be split into separate regions.
- **Dimension notes**: for every dimension entry, a concrete `notes` sentence is required describing the actual visual appearance (specific colours, paddings, weight hierarchy); restating the variant name verbatim is prohibited.
- **Colour palette**: the global palette must name four roles— `surface` (background), `primary` (brand colour), `accent` (highlight), `fg` (foreground text)—each as a hex value observed in the image.
- **Design caption**: each region must include a `design_caption` of 3–5 sentences describing its visual design holistically (colour usage, typographic hierarchy, density, decoration), without restating per-dimension variant names.

Output. A single JSON object with fields `global_caption`, `global` (palette, typography, decorations), and a `regions` array. Each element of `regions` contains the region type, title, reading order, content caption, all 9 per-region dimension tokens (each a `{value, notes}` pair), and a free-form `design_caption`. No markdown fences, no commentary, no think blocks.

B.2.2 Prompt 2: HTML Reconstruction in Derendering

Used in Stage B of the derendering pipeline to convert the layout and design IR into a pixel-faithful HTML/CSS reproduction of the poster.

System prompt. *"You are an HTML/CSS reconstruction expert. Your goal: make the rendered HTML look **identical** to the attached reference image."*

User prompt. The MLLM receives five inputs: (1) the **reference poster image** (visual ground truth); (2) a **layout JSON** containing, for every detected region, its bounding-box coordinates (xyxy, in pixels), semantic class label (title / plain text / figure / table / formula / ...), OCR-extracted text content, and OCR-measured font size (computed as crop_height$/n_{\text{lines}} \times 0.8$); (3) the **design specification** produced by Prompt 1, comprising the global colour palette, typography family, page-level decoration types, and per-region design tokens (`composition`, `visual_anchor`, `flow`, `interaction`, `density`, `color_scheme`, `typography`, `emphasis`, `decoration`) each paired with a free-form `notes` sentence; (4) **dimension→CSS hints**: for each enum variant the recommended CSS pattern (e.g. `solid_carrier`⇒a `position:absolute` background `<div>` at `z-index:0` with a coloured title strip; scaffold content divs raised to `z-index:2`); (5) **font-size hints**: per-block target `font-size` values; entries tagged "[measured by OCR]" are exact targets and must be reproduced precisely in CSS.

Key rules.

- **Geometry lock**: every block's `left`/`top`/`width`/`height` is immutable; the MLLM must not add, remove, or reorder scaffold `<div>` elements, and must not alter canvas dimensions.

- **Text fidelity**: for each text-like block, the MLLM must read the *exact text* from the reference image—reproducing every word, symbol, capitalisation, and line break. Mathematical expressions must be preserved using Unicode symbols, `<sup>`/`<sub>`, and CSS alignment; text must never overflow its bounding box.
- **Style matching**: the MLLM must match font family (sans-serif / serif / monospace), font size (using the OCR-measured value as target), font weight and style, text colour, alignment, and line spacing as observed in the reference image.
- **Region backgrounds**: when a region's `interaction` token is not `flat_inline` or its `decoration` token is not `border_none_shadow`, extra background `<div>` elements must be injected inside the canvas (z-index 0 for the outer coloured frame; z-index 1 for an inner white box if noted by the design caption) enclosing that region's blocks; scaffold block divs are then raised to z-index 2.
- **Visual reproduction**: all backgrounds, gradients, borders, divider lines, decorative shapes, header bars, and shadows must be reproduced using only CSS (`background`, `border`, `box-shadow`, `border-radius`, `linear-gradient`, `::before`/`::after`); no external images or scripts are permitted.
- **Design spec precedence**: the design specification from Prompt 1 takes precedence over the MLLM's own interpretation of the reference image whenever they conflict.

Output. Complete HTML from `<!doctype html>` to `</html>`. No markdown fences, no explanations, no `<script>` tags, no external URLs.

B.2.3 PROMPT 3: PAPER PARSER (PARSERAGENT)

Used in Stage 1 to condense a parsed paper into poster-ready content.

System prompt. *"You are an expert academic paper analyst. Your task is to read parsed LaTeX paper content and reorganize it into a concise, well-structured JSON format suitable for academic poster generation."*

User prompt. Provides the parsed metadata (title, authors, affiliations), abstract, and each section's content (capped at 3,000 characters per section, with subsections capped at 1,500 characters). Instructions:

1. Review all sections and determine importance for a poster.
2. Condense each section to 50–200 words.
3. Preserve key technical details, results, and conclusions.
4. Ensure coverage: Title & Authors, Introduction/Motivation, Method, Results, Conclusion.
5. Keep content factual—do not add information not in the paper.
6. Create short section titles (1–4 words).

Output. JSON with `meta` (poster_title, authors, affiliations) and `sections[]` (title, content, importance: `high/medium/low`).

B.2.4 PROMPT 4: POSTER PLANNER (PLANNERAGENT)

Used in the baseline (non-template) pipeline to plan poster structure. Not used in the template-filling pipeline where the template's layout is preserved.

System prompt. *"You are an expert academic poster designer. You plan the structure and layout of academic conference posters."*

User prompt. Provides poster dimensions, paper content (sections with importance, first 500 characters each), available figures/tables with captions. If a reference image is provided, the prompt instructs the planner to replicate its column structure, header style, section arrangement, figure placement, and overall flow. Instructions:

1. Select 5–8 sections for the poster.

2. Assign figures/tables to relevant sections.
3. Decide column layout (matching reference if provided).
4. Assign content priority (space weight per section).

Output. JSON with `layout` (num_columns, orientation, color_scheme) and `poster_sections[]` (title, content_source, figure_id, table_id, space_weight, position_hint).

B.2.5 PROMPT 5: HTML GENERATOR (GENERATORAGENT)

Used in the baseline pipeline for from-scratch HTML generation. Not used in the template-filling pipeline.

System prompt. *"You are an expert HTML/CSS developer specializing in academic poster design. You create beautiful, professional conference posters as self-contained HTML files."*

User prompt. Provides poster dimensions, the layout plan JSON, and poster section content with figure placeholders (`FIGURE_PLACEHOLDER_N`). The prompt supports three style-guidance modes: (a) image + CSS spec from StyleExtractor; (b) image only; (c) HTML template source. Design requirements include: cohesive color scheme, prominent header, N-column grid, clear section headers, readable font sizes (title: 42–48px, headers: 28–32px, body: 18–22px), and CSS Grid/Flexbox layout. 11 critical rules enforce dimension accuracy, placeholder preservation, visual hierarchy, and style matching.

Output. Complete self-contained HTML.

B.2.6 PROMPT 6: CONTENT MAPPER (MAPPINGAGENT)

System prompt. *"You are an expert academic poster designer. Your task is to map the content of a research paper into the slots of an existing poster template. You do NOT design layout or CSS—the template's visual design is fixed. You only decide WHAT TEXT goes into WHICH SLOT."*

User prompt. Provides four input blocks:

1. **Paper content**: metadata, abstract, summarized sections (with importance), figures (ID + caption + paths), tables, pre-rendered equations/tables, and optionally raw section content (up to 2,000 chars/section) for additional material.
2. **Template slots**: the slot summary generated by the template parser, listing each slot's `block_id`, type, label, position, dimensions, font size, and estimated character capacity (`chars_per_line` × `max_lines`).
3. **Mapping rules**: 4 action types with detailed constraints (see below).
4. **Content distribution rules**: ensuring even distribution across columns.

Mapping action constraints.

- `replace_text`: text **must** fit slot capacity; use bullet points ("•") for body text; section headers should be 1–3 words. **Every text-like slot must use this action.**
- `replace_image`: only for image-like slots; specify `figure_id`; choose figures matching the slot's aspect ratio. **Use at least 4–6 different paper figures.**
- `replace_rendered`: for image slots where an equation or table is appropriate; specify `rendered_key` (e.g., `equation_1`, `table_1`).
- `keep`: **only** for purely decorative elements (borders, dividers, background panels). **Never** for image slots containing academic content.
- `clear`: **only** for image-like slots with no matching figure. **Never** for text-like slots.

Positional conventions. First title slot → paper title; author slot → authors/affiliations; largest text slots → most important sections; top-row → introduction/motivation; middle → methods; bottom → results/conclusion.

Overflow handling. If more text slots than paper sections, the agent must: split long sections, expand bullet points, add content from abstract/related work, or create "Key Findings"/"Contributions"/"Future Work" sections.

Output. JSON: `{slot_mappings: [{block_id, action, new_text?, figure_id?, rendered_key?}]}`.

B.2.7 PROMPT 7: CONTENT ADAPTER (ADAPTERAGENT)

Used after programmatic filling to refine text content length.

System prompt. *"You are an expert academic poster editor. You refine text content inside an HTML poster template. You MUST NOT change any CSS, layout, positions, colors, font sizes, or DOM structure. You may ONLY edit the text inside `<div class="content">...</div>` tags."*

User prompt. Provides the filled HTML poster (with base64 images stripped and replaced by `BASE64_STRIPPED_N` placeholders to reduce size from ∼500KB to ∼30KB), the paper's metadata and key sections (up to 300 chars each). The agent's 5 tasks:

1. Shorten text that is too long and might overflow its slot.
2. Expand text that is too short and wastes space.
3. Format with bullet points and proper line breaks.
4. Ensure author/affiliation accuracy.
5. Ensure section headers are clear and concise.

Strict prohibitions (enforced in prompt).

- Do NOT modify `<style>` tags or any CSS properties.
- Do NOT add/remove `<div>` elements or change `id`/`class`/`style` attributes.
- Do NOT change `<img> src` attributes or `BASE64_STRIPPED_N` placeholders.
- ONLY modify text between `<div class="content">` and `</div>`.

Output. Complete HTML file with only text content changes. Placeholders are restored to real base64 data URIs after refinement.

B.2.8 PROMPT 8: VISUAL REVIEWER (REVIEWERAGENT)

System prompt. Defines the agent as an expert visual reviewer with expertise in typography, image layout, visual hierarchy, and academic poster aesthetics. Explicitly lists 8 hard constraints:

- Must NOT change layout geometry (left, top, width, height).
- Must NOT change color schemes, backgrounds, or gradients.
- Must NOT change DOM structure (no add/remove elements).
- Must NOT hide elements (`display:none`, `visibility:hidden`, `opacity:0`).
- Must NOT truncate text or shrink font sizes.
- Must NOT add `overflow:hidden`, `white-space:nowrap`, or `text-overflow:ellipsis` to text blocks.
- CAN adjust: `line-height`, `word-break`, `overflow-wrap`, `z-index`, `padding`.
- CAN set `overflow:hidden` on image blocks only.

Critical context: the poster content is *intentionally* different from the original template; content mismatch must not be reported as an issue.

User prompt. Provides: poster dimensions, slot information summary, previous round fixes (if any), and a structured 5-item checklist:

1. **Text overflow** → `word-wrap:break-word`, `overflow-wrap:break-word`, tighter `line-height`. Never truncate, never `overflow:hidden`, never shrink font-size.
2. **Text too small** → increase `font-size`. Never decrease.

3. **Image display** → `img::object-fit:contain`, `img::max-width:100%`, `img::max-height:100%`.
4. **Overlapping elements** → adjust `z-index` only.
5. **Poor readability** → adjust `line-height` (not font-size).

Non-issues (must not be reported). Content mismatch with original template; broken/placeholder images; intentionally empty blocks.

Output format. JSON:

```
{
  "overall_score": <1-10>,
  "issues_found": <int>,
  "review_notes": "<1-2 sentence assessment>",
  "fixes": [{
    "block_id": "blk-XXXX",
    "issue_type": "<text_overflow|text_too_small|
        image_overflow|overlap|readability|...>",
    "description": "<what is wrong>",
    "css_fixes": {"<property>": "<value>"},
    "truncate_text": false
  }]
}
```

Scoring rubric.

- 9–10: Near perfect—clean, readable, professional, all content fits within blocks.
- 7–8: Good—minor issues like slightly tight text, but overall professional.
- 5–6: Acceptable—some visible overflow or sizing issues, but poster is usable.
- 3–4: Poor—major overflow, unreadable text, or significant layout problems.
- 1–2: Unusable—severe rendering failures.

CSS fix conventions. Property names in kebab-case (`font-size`, not `fontSize`); values include units (`16px`, `1.2`); image-specific fixes use `img::` prefix (e.g., `"img::object-fit": "contain"`).

B.3 Programmatic Filling Details

Text replacement. For each `replace_text` action, the filler locates the target block by regex matching on `<div id="blk-XXXX">` and replaces the inner HTML of its `<div class="content">` child. Existing HTML tags within the new text (e.g., `<b>`, `<i>`, `<sub>`, `<sup>`) are preserved to support inline formatting.

Image replacement. For `replace_image`, the filler resolves the figure ID through the figure registry, attempting multiple path variants (with/without extension, with/without directory prefix such as `fig/`, `figures/`, `images/`) to handle diverse LaTeX path conventions. The resolved base64 data URI replaces the `src` attribute.

Equation and table rendering. LaTeX equations are rendered to PNG via MathJax (for inline and display math) and pdflatex (for complex table environments). Rendered images are stored as base64 data URIs in a separate equation/table registry and injected via the `replace_rendered` action.

Safety passes. After all mappings are applied, two safety passes run:

1. **Unmapped image clearing**: any image-like slot not covered by the mapping receives `display:none` to avoid showing template placeholder images from the original poster.
2. **Stale asset sweep**: any `<img>` element whose `src` still references `assets/*` (template-local crops) is hidden, preventing residual template content from leaking into the generated poster.

Font size optimization. After filling, text slot font sizes are dynamically adjusted based on the new content length relative to the slot's capacity. This ensures that the text fills each slot comfortably without overflow or excessive whitespace.

B.4 CSS SAFETY GUARD IMPLEMENTATION

The ReviewerAgent's CSS fix application includes a multi-layer safety mechanism to prevent destructive modifications:

Table 6: CSS safety guard rules. Any VLM-proposed fix matching these conditions is silently blocked.

Rule	Blocked Property	Blocked Values
Element hiding	`display`	`none`, `hidden`, `0`, `collapse`
Visibility hiding	`visibility`	`hidden`
Opacity hiding	`opacity`	`0`
Font shrinking	`font-size`	Any value $<$ template original
Text clipping	`overflow` (text blocks)	`hidden`, `clip`
No-wrap	`white-space` (text blocks)	`nowrap`, `pre`
Ellipsis	`text-overflow` (text blocks)	`ellipsis`, `clip`

For each proposed CSS fix, the guard: (1) checks the property–value pair against the forbidden list; (2) for font-size changes, retrieves the original template font size and rejects any decrease; (3) distinguishes text-like and image-like blocks, applying text-hostile property blocking only to text blocks (image blocks *may* receive `overflow:hidden` to contain images within their bounding box).

B.5 POST-TRAINING DYNAMICS

Figures 8 and 9 summarize the two-stage post-training dynamics. During SFT, training and validation loss decrease as next-token accuracy improves; validation loss and accuracy approach 0.24 and 92.2%, respectively, as the learning rate decays. During RL, the mean total reward rises from approximately 0.40 to 0.69 over 300 steps. The model-based reward shows the larger increase, while the rule-based reward remains high after an initial rise. Completion length decreases from roughly 5.1k to 4.2k tokens. These curves characterize training-objective optimization rather than held-out poster quality, which is evaluated separately in the main experiments.

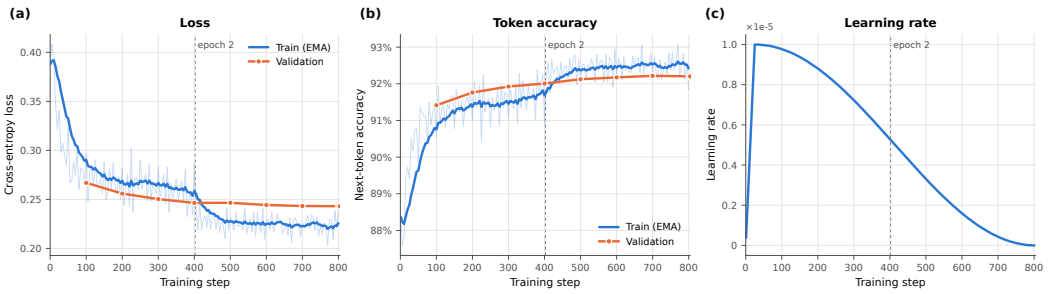

Figure 8: Supervised fine-tuning dynamics of Qwen3.8-27B: (a) cross-entropy loss, (b) next-token accuracy, and (c) learning rate. In (a–b), pale blue traces show per-step training measurements, bold blue curves show their exponential moving averages, and orange markers show validation measurements. Dashed lines mark the start of the second epoch.



Figure 9: Reinforcement-learning dynamics after SFT: (a) mean total reward, $R = 0.4R_{\text{rule}} + 0.6R_{\text{model}}$; (b) rule-based and model-based reward components; and (c) generated completion length. Light traces show per-step measurements, while bold curves show smoothed trends.

B.6 Ablation on RL Rewards

Table 7 compares the rule-based reward, the model-based reward, and their combination. The combined configuration outperforms both single-reward variants on LPIPS, DISTS, FID, and FDiD. Compared with using the model-based reward alone, it reduces FID from 25.58 to 23.42 and FDiD from 313.34 to 306.94, while DINOv2-CosDist decreases slightly from 0.2784 to 0.2735. The combined objective thus improves distributional human-likeness and two of the three diversity measures.

Table 7: Ablation of RL reward components on Qwen3.8-27B. Rule-based and Model-based denote DOM-rule and VLM-based rewards, respectively. Best results are in **bold**; second-best results are underlined.

Setting	Rewards		Diversity			Human-likeness	
	Rule-based	Model-based	LPIPS↑	DINOv2-CosDist↑	DISTS↑	FID↓	FDiD↓
Base	–	–	0.6284	0.2579	0.2680	32.44	465.00
SFT	–	–	0.6599	**0.2784**	0.2892	28.00	326.24
RL	✓	✗	0.6554	0.2733	0.2834	27.63	323.61
RL	✗	✓	0.6735	**0.2784**	0.2867	25.58	313.34
RL (ours)	✓	✓	**0.6925**	0.2735	**0.2944**	**23.42**	**306.94**

C Evaluation Details

C.1 Metrics

C.1.1 Diversity

Given the generated poster set $\{\hat{P}_i\}_{i=1}^{N}$ produced by a method on the N-sample test set, we compute each diversity metric pairwise over all $\binom{N}{2}$ pairs and report the average. Higher pairwise distances indicate more structurally and perceptually varied outputs across different input papers.

LPIPS (Zhang et al., 2018). Learned Perceptual Image Patch Similarity measures perceptual distance via weighted ℓ_2 distance between deep feature activations of a pretrained AlexNet. Higher mean pairwise LPIPS indicates that posters look more visually distinct to human perception across the generated set.

DINOv2-CosDist. We extract a global feature embedding from each poster using the pretrained DINOv2 (Oquab et al., 2023) ViT-L/14 encoder and compute the cosine distance $(1 - \cos)$ between every pair of embeddings. As DINOv2 is trained via self-supervision to capture high-level semantic and stylistic structure, larger DINOv2-CosDist values indicate that the generated posters span a wider range of design styles and visual themes in the model's semantic space, providing a strong indicator of stylistic diversity beyond pixel-level variation.

DISTS (Ding et al., 2020). Deep Image Structure and Texture Similarity extracts multi-scale VGG features and measures both structural and textural dissimilarity in a distortion-tolerant manner. Higher mean pairwise DISTS indicates greater deep-feature diversity across the generated set.

C.1.2 Human-Likeness

To measure the distributional gap between generated posters and real human-created designs, we compute the Fréchet distance in different feature spaces. Given the feature representations of real posters (reference distribution r) and generated posters (generated distribution g), the Fréchet distance d^2 is defined as:

$$d^2 = \|\mu_r - \mu_g\|^2 + \mathrm{Tr}\Big(\Sigma_r + \Sigma_g - 2\sqrt{\Sigma_r \Sigma_g}\Big) \tag{4}$$

where μ and Σ denote the mean vector and covariance matrix of the respective feature distributions. To capture different aspects of poster fidelity, we employ two metric variants by changing the underlying feature extractor:

- **Fréchet Inception Distance (FID) (Heusel et al., 2017):** Extracts 2048-dimensional features from the `pool3` layer of an ImageNet-pretrained InceptionV3 (Szegedy et al., 2016). FID captures general image synthesis quality but is relatively insensitive to document-specific layouts.
- **Fréchet DINOv2 Distance (FDiD):** Extracts 768-dimensional features from the `pooler_output` of a self-supervised `dinov2-base` model (Oquab et al., 2023). By utilizing self-supervised vision transformer representations, FDiD is highly sensitive to visual structure, spatial arrangement, and layout geometry.

Lower values across both metrics indicate that the generated posters are distributionally closer to real human designs.