Paper2Poster: Towards Multimodal Poster Automation from Scientific Papers

[1,3]Wei Pang*, [2]Kevin Qinghong Lin*✉, [1]Xiangru Jian*, [1,3]Xi He✉, [2]Philip Torr
[1] University of Waterloo [2] University of Oxford [3] Vector Institute
Project Page: https://paper2poster.github.io

Abstract

Academic poster generation is a crucial yet challenging task in scientific communication, requiring the compression of long-context interleaved documents into a single, visually coherent page. To address this challenge, we introduce the first benchmark and metric suite for poster generation, which pairs recent conference papers with author-designed posters and evaluates outputs on (i) Visual Quality—semantic alignment with human posters, (ii) Textual Coherence—language fluency, (iii) Holistic Assessment—six fine-grained aesthetic and informational criteria scored by a VLM-as-judge, and notably (iv) **PaperQuiz**—the poster's ability to *convey core paper content as measured by VLMs answering generated quizzes*. Building on this benchmark, we propose **PosterAgent**, a top-down, visual-in-the-loop multi-agent pipeline: the *(a) Parser* distills the paper into a structured asset library; the *(b) Planner* aligns text–visual pairs into a binary-tree layout that preserves reading order and spatial balance; and the *(c) Painter–Commenter* loop refines each panel by executing rendering code and using VLM feedback to eliminate overflow and ensure alignment. In our comprehensive evaluation, we find that GPT-4o outputs—though visually appealing at first glance—often exhibit noisy text and poor PaperQuiz scores, and we find that reader engagement is the primary aesthetic bottleneck, as human-designed posters rely largely on visual semantics to convey meaning. Our *fully open-source* variants (*e.g.,* based on the Qwen-2.5 series) outperform existing 4o-driven multi-agent systems across nearly all metrics, while using $\mathbf{87}\%$ **fewer tokens**. It transforms a 22-page paper into a finalized yet editable '`.pptx`' poster — all for just $\$0.005$. These findings chart clear directions for the next generation of fully automated poster-generation models. The code and datasets are available at https://github.com/Paper2Poster/Paper2Poster.

1 Introduction

Academic posters play a pivotal role in scientific communication, enabling rapid dissemination of key findings at conferences where attendees have only minutes to grasp core insights from the full papers. Despite significant progress in automated slide generation – with systems such as PPTAgent [37] and D2S [29] pioneering text-to-slide pipelines – poster creation [33, 30, 3] remains an underexplored and substantially more challenging task. Unlike slide decks, which distribute content across multiple, single-message slides, academic posters must condense an entire paper into a single, visually coherent page. This requires **(i)** handling a much longer multi-modal context [24], **(ii)** tightly interleaving text and graphics to convey complex ideas at a glance [33, 3], and **(iii)** respecting stringent spatial constraints to avoid text overflow or layout collapse [10, 30]. These factors make VLM- or LLM-only approaches insufficient: without explicit visual feedback like humans, it is difficult to reason about spatial layouts, maintain logical flow within a confined canvas, ensuring legibility and aesthetic.

To systematically evaluate poster generation, we propose the **Paper2Poster** Benchmark, the first benchmark and metric suite for this novel task. Our benchmark comprises recent conference papers

*Equal contribution. ✉ Corresponding to: xihe@uwaterloo.ca, kevin.qh.lin@gmail.com

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Track on Datasets and Benchmarks.

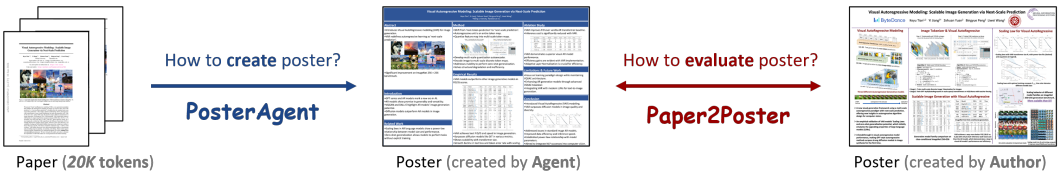


Figure 1: **Overview of this work.** We address two core challenges in scientific poster generation: **Left: How to create a poster from a paper**—we propose **PosterAgent** (Sec. 4), a framework that transforms long-context scientific papers (20K+ tokens) into structured visual posters; and **Right: How to evaluate poster quality**—we introduce the **Paper2Poster** benchmark (Sec. 3), which enables systematic comparison between agent-generated and author-designed posters.

paired with author-designed posters, along with a human-and-model evaluation protocol that measures **(i)** Visual Quality — how well the generated poster aligns visually with the human-designed version. **(ii)** Textual Coherence — the clarity and fluency of the poster's language. **(iii)** Holistic Assessment — the overall aesthetic and informational quality, rated across six fine-grained dimensions by VLM as Judge. Notably, **(iv) PaperQuiz** — motivated by the poster's role as a bridge between authors and readers, this metric evaluates *how effectively the poster alone conveys core paper content* by simulating diverse reader comprehension using VLMs to answer questions derived from the paper.

To tackle multimodal context compression in Paper2Poster, we introduce **PosterAgent**, a multi-agent framework that first globally organizes document content and then performs panel-level refinements—while weaving visual feedback into every stage. Starting with the **Parser**, we ingest the full paper PDF and transform it into an asset library of section-level text summaries and extracted figures and tables. Next, the **Planner** semantically matches each synopsis to its corresponding visual asset and generates a binary-tree layout, allocating panels by estimated content length while preserving reading order and spatial balance. Finally, the **Painter–Commenter** loop refines each panel: the Painter distills section-figure pairs into concise bullet points and renders draft panels via `python-pptx` code, and the Commenter—a VLM with zoom-in reference prompts—provides targeted feedback to correct text overflow and spatial alignment. This *top-down, visual-in-the-loop* design produces concise, coherent posters without manual tuning.

Using Paper2Poster, we comprehensively evaluate human-designed (oracle) posters, state-of-the-art generative models (*e.g.,* GPT-4o), and multi-agent solutions, revealing several key insights: *(i)* GPT-4o's outputs, though visually appealing at first glance, suffer from noisy or incoherent text, yielding high perplexity and poor PaperQuiz performance; *(ii)* VLM-based judging shows the primary aesthetic bottleneck is Engagement rather than informational content, since human posters convey meaning predominantly through visual semantics; *(iii)* PaperQuiz proves a reliable metric—VLM reader scores correlate closely with human evaluations, and more capable VLMs achieve higher scores on well-designed posters; and *(iv)* our Paper2Poster pipeline, built on a *fully open-source* toolbox (*e.g.,* `Qwen-2.5-VL-7B`), surpasses existing GPT-4o–based multi-agent approaches on nearly all metrics while consuming **87% fewer tokens**. Our findings illuminate pathways for the next generation of models and agent systems aimed at fully automated poster generation.

2 Related Work

2.1 Visual Design Automation

Recent advances in multi-modal learning have driven significant progress in automating visual design tasks. These tasks commonly fall into two broad categories: **(i) Text-rich Image Generation.** Tasks such as poster generation [3, 17, 11, 33] have greatly benefited from diffusion-based approaches [11, 10, 31], which enable the synthesis of detailed visuals conditioned on natural language descriptions. However, ensuring the quality and fidelity of embedded textual content via an end-to-end pixel generative model remains a major challenge, as generated text at the pixel level appears blurry and hard to read. **(ii) Complex Visual Layouts.** Tasks like website designing [7, 27, 16, 23] or slide generation [37, 2, 8, 18, 26, 29] involve intricate visual structures and require integrating diverse components. To handle such complexity, mainstream approaches [37, 5] often employ agentic workflows that rely heavily on code generation and tool usage to assemble complete visual outputs. In contrast, our Paper2Poster addresses a more demanding yet highly practical setting: scientific visual design based on academic papers. This involves *long-context, interleaved multi-modal, inputs and outputs*, posing substantial challenges in both effectiveness and computational efficiency.

2.2 Vision-Language Agents

Recent progress has revealed the promising potential of LLMs beyond pure language understanding. Techniques such as ReAct [36, 35] have demonstrated that LLMs can serve as autonomous agents, capable of solving complex tasks through step-by-step reasoning and dynamic interaction via coding [32, 34], API function calling [25, 15], or UI interface interaction [13, 22, 19]. Despite these advances, general-purpose agents still struggle with professional tasks [12] as they require serious, accurate interaction and domain-specific knowledge. One closely related application is slide automation [5, 37], where agents translate brief textual queries into executable Python code (*e.g.,* via `python-pptx`) to render presentation slides. However, our Paper2Poster setting is significantly more challenging: instead of a text prompt, we take full-length academic papers as inputs and generate compact, well-structured posters as output. This novel task requires careful design of both evaluation metrics and an effective, practical automation workflow.

3 Paper2Poster Benchmark

3.1 Task Definition

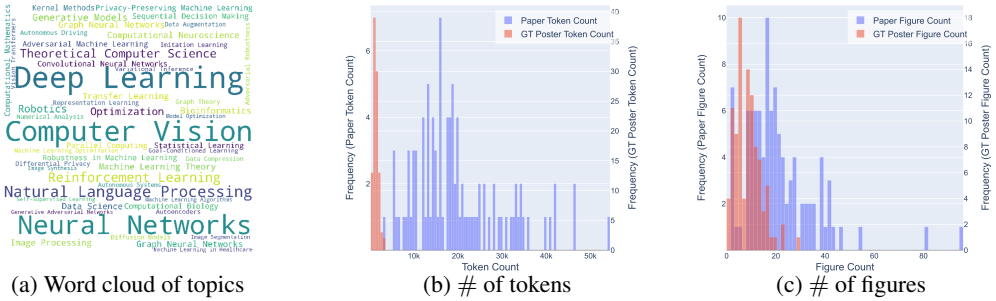
Given a scientific paper composed of interleaved text, figures, and tables, the goal is to automatically generate a single-page academic poster that faithfully conveys the paper's core content in a visually coherent and spatially efficient format. This task presents several unique challenges: ***a*. Long-Context Long-Horizon Task**: Scientific papers span multiple pages and thousands of words. Summarizing key insights while preserving coherence demands hierarchical understanding and selective abstraction. The complexity further necessitates long-horizon reasoning and multiple iterative interactions, making the task especially challenging. ***b*. Interleaved Multimodal Inputs**: Papers integrate numerous figures, tables, and charts, each semantically linked to the surrounding text. Successful poster generation demands the ability to extract, interpret, and align these multimodal elements in a contextually appropriate manner. ***c*. Layout-Aware Multimodal Outputs**: Unlike tasks focused solely on text (*e.g.,* blog) or vision, poster generation requires producing interleaved text–image outputs within a constrained spatial layout. This necessitates joint reasoning over language, visual content, and layout to prevent overflow, imbalance, and logical misalignment.

3.2 Data Curation

Data Source. We focus exclusively on AI papers for three key reasons: (1) they are relatively recent and undergo rigorous peer review, ensuring high scientific quality; (2) they offer diverse content across subfields—such as image-rich computer vision, text-centric NLP, and theory papers with numerous equations—providing a broad range of input modalities. To support this, we adopt the PosterSum dataset [24], which contains a large collection of paper–poster pairs from recent AI conferences including ICML, NeurIPS, and ICLR (2022–2024). We specifically use the test split to reduce the risk of overlap with training data.

Diverse Sampling. Based on the initial candidate set, we apply two filtering criteria to curate high-quality data: (1) **Length Control**: We deliberately include longer papers, including supplementary material, selecting PDFs that exceed 15 pages and extend up to 50 pages. (2) **Latest Version**: We manually retrieve the most recent PDF version for each paper to ensure the dataset reflects final camera-ready submissions. From the filtered set, we construct the final Paper2Poster dataset consisting of 100 paper–poster pairs, stratified by publication year to ensure temporal balance: 33 pairs from 2022, 33 from 2023, and 34 from 2024. To further enhance diversity, we also stratify by source venue—selecting 35 papers from NeurIPS, 37 from ICML, and 28 from ICLR, ensuring broad coverage across these leading conferences.

Data Statistics. Overall, Paper2Poster comprises 100 paper-poster pairs spanning 280 distinct topics across domains such as Computer Vision (19%), Natural Language Processing (17%), and Reinforcement Learning (10%), ensuring comprehensive coverage across subfields. As illustrated in Fig. 2 (a-b), the input papers contain an average of 12155.7 words across 22.6 pages, amounting to approximately 20370.3 tokens, with an average of 22.59 figures per paper. In Fig. 2 (c-d), the corresponding author-designed posters include an average of 774.1 words (1416.2 tokens) and 8.7 figures. This reflects a textual compression ratio of approximately 14.4× and a figure reduction ratio of about 2.6× from paper to poster.



(a) Word cloud of topics (b) # of tokens (c) # of figures

Figure 2: **Data Statistics of Paper2Poster.** (a) Word cloud illustrating the diversity of research topics. (b) Textual Token statistics and Figure count statistics for input papers *vs.* posters provided by authors. Overall, these statistics highlight that Paper2Poster is a multimodal context compression task, requiring effective abstraction of both textual and visual content.

3.3 Evaluation Metrics

To systematically measure the quality of generated posters, we establish a comprehensive evaluation framework that covers four essential dimensions as shown in Fig. 3 (left): **(i) visual quality**, **(ii) textual coherence**, **(iii) quality assessment via VLM** (*i.e.,*VLM-as-judge), and notably our proposed **(iv) PaperQuiz** which measures how effectively the poster conveys the paper's core knowledge.

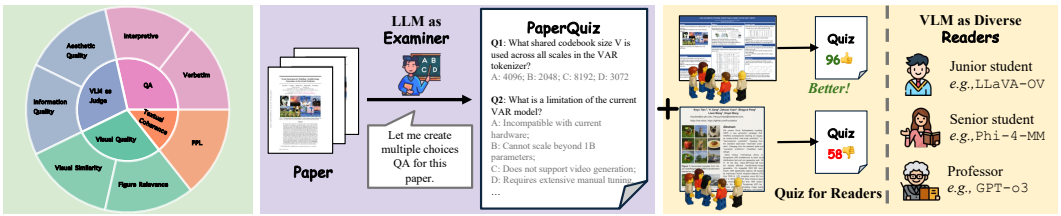


Figure 3: **Left**: Overview of the evaluation framework in Paper2Poster. **Middle**: We automatically generate multiple-choice questions from each paper using an LLM (o3), forming the our **PaperQuiz** evaluation. **Right**: In PaperQuiz, we simulate multiple reader by allowing VLMs—representing different expertise levels (*e.g.,* student, professor)—to read each generated poster and answer the quiz. The poster that achieves the highest average score is considered the most effective in conveying the paper's content.

(i) Visual Quality. The visual presentation of a poster directly impacts reader comprehension and engagement. To evaluate visual quality from both global and local perspectives, we employ two metrics: (1) We measure "*Visual Similarity*" between the generated and the author-designed posters as ground-truth using CLIP image embeddings. This metric captures high-level visual–textual correspondence to assess whether outputs are truly "poster-like" rather than article-like layouts, though it is not a direct measure of aesthetic quality. This approach is favored over traditional distribution-based metrics (such as FID used in prior works [5, 37]), as it assesses instance-level semantic consistency. (2) We measure "*Figure Relevance*" by computing the average CLIP similarity between figures and their corresponding text sections in the original paper. This metric ensures figures are contextually appropriate and effectively integrated, assigning zero relevance to posters lacking visual content. For both metrics, we employ AltCLIP [4] due to its robustness in handling longer sequences alignment. We complement CLIP with aspect-level VLM-as-Judge evaluation (see below) to capture fine-grained visual quality that CLIP may not fully capture. Detailed definition of both metrics can be found in Appendix F.1.

(ii) Textual Coherence. Clear and fluent text is essential for poster readability and comprehension. We therefore quantify textual coherence by computing the standard "*Perplexity*" (PPL) of the entire poster text under `Llama-2-7b-hf`. Lower PPL indicates more predictable, coherent language. Importantly, PPL assesses *fluency and local coherence* rather than semantic similarity to a reference, making it well-suited for our abstractive poster generation task where content should be reorganized and compressed rather than copied. A detailed definition is provided in Appendix F.2.

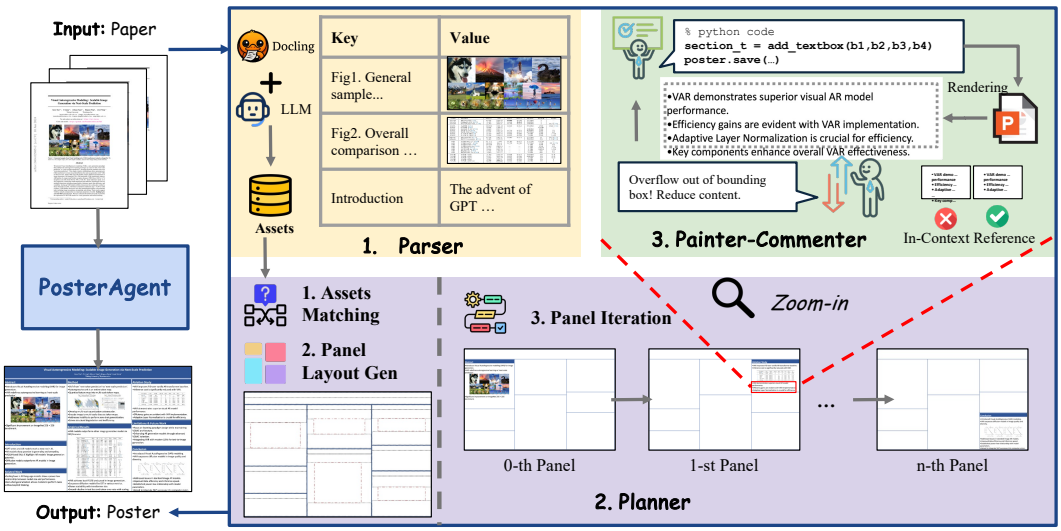


Figure 4: **Illustration of the PosterAgent pipeline.** Given an input paper, PosterAgent generates a structured academic poster through three modules: **1. Parser:** Extracts key textual and visual assets using a combination of tools and LLM-based summarization, resulting in a structured asset library. **2. Planner:** Matches assets and arranges them into coherent layouts, iteratively generating panels with a *zoom-in* operation. **3. Painter–Commenter:** The Painter generates panel-level bullet-content along with executable code, and renders the visual output, while the Commenter—a VLM with in-context reference—provides feedback to ensure layout coherence and prevent content overflow.

(iii) Holistic Assessment (VLM-as-Judge). To evaluate overall poster effectiveness in fine-grained dimension, we prompt a VLM (*e.g.,* GPT-4o) as an automated judge by outputting score (1–5). For each poster image, the model assigns 6 criterion-level scores: 3 under "*Aesthetic Score*"—{Element Quality, Layout Balance, Engagement}, and 3 under "*Information Score*"—{Clarity, Content Completeness, Logical Flow}. This direct, image-centric evaluation preserves fidelity to both visual design and content, while also capturing informativeness. It provides fine-grained feedback to guide future poster design. Full prompt templates and scoring protocols are detailed in Appendix F.3.

(iv) PaperQuiz. Given the poster's central role in communicating the content of its source paper—serving as a bridge between authors and readers—we design an evaluation protocol that simulates this communication scenario. As shown in Fig. 3 (middle), each paper PDF is first submitted to o3 as examiner to generate 100 multiple-choice questions per paper: 50 *verbatim* questions (directly answerable from the text, spanning 13 content aspects) and 50 *interpretive* questions (targeting high-level comprehension across 10 conceptual dimensions). Next, as illustrated in Fig. 3 (right), we present each poster image to six VLMs (both open- and closed-source), simulating a range of reader standards from casual to expert. These models then answer the quiz based solely on the poster content. By comparing their quiz scores across different poster variants, we identify which poster best conveys the original paper content. Given that a poster is a visual medium rather than plain text like a note, we further adjust the raw Quiz scores $\mathbf{s}_r \in [0, 100]$ by incorporating a length-based penalty, resulting in a penalized score $\mathbf{s}_a \in [0, 200]$:

$$\mathbf{s}_a = \mathbf{s}_r \left(1 + \frac{1}{\max(1, L/W)}\right),$$

where L denotes the total text length of the poster, and W is the median text length of human-designed (ground-truth) posters. This penalty function is designed with three goals: (i) discourage overly long posters ($L \gg W$ yields $\mathbf{s}_a \to \mathbf{s}_r$, losing the bonus), (ii) avoid harsh punishment (as $L \to \infty$, $\mathbf{s}_a$ remains at $\mathbf{s}_r$, not approaching zero), and (iii) prevent rewarding extreme brevity (when $L \leq W$, the bonus is capped at $\mathbf{s}_a = 2\mathbf{s}_r$, so further shortening provides no additional gain). By anchoring the penalty to human-designed poster lengths, we ensure that posters are neither excessively verbose nor sacrifice informative content for brevity. Further details on metric design, question curation, evaluation workflow, and scoring procedures can be found in Appendix F.4.

4 PosterAgent

Overview. Identifying the challenges posed by the Paper2Poster, we formulate it as a problem of multimodal context compression, and introduce PosterAgent, a multi-agent pipeline that adopts a "*Top-down*" design philosophy: it first globally restructures the entire document into concise, coherent sections, followed by local refinements for fine-grained, panel-level control. As shown in Fig. 4. The pipeline consists of three key components: **1. Parser:** Extracts key textual and visual content by tools and LLM-based summarization to build an asset library. **2. Planner:** Aligns assets and arranges them into coherent layouts, generating panels iteratively with a *zoom-in* mechanism. **3. Painter–Commenter:** The Painter produces panel-level bullet points and executable code for rendering, while a VLM as Commenter—ensures layout coherence and avoids overflow.

4.1 Parser: *global organization*

Given a paper, the first step is to *globally* organize the information into a structured format to support subsequent processing. This is handled by the **Parser**, which performs a coarse-grained compression by ingesting the raw PDF and producing an **asset library** across two modalities: **(1) Text assets** that capture the document hierarchy like human first glance focus on section heading—each key is a section heading and the associated value a paragraph-level synopsis; **(2) Visual assets** built in parallel, where figure or table captions serve as keys and the extracted image files are stored as values. We leverage MARKER[21] and DOCLING[14] to convert each page into Markdown, which is then processed by an LLM to generate a structured, JSON-like outline. This transformation compresses the raw text into a compact asset library that preserves essential semantics while significantly reducing size, enabling more efficient downstream iteration and layout generation.

4.2 Planner: *local organization*

With the visual and text assets collected by the Parser, the next step is to select the relevant content and begin constructing the poster. Rather than generating the entire poster in one shot, we emphasize the importance of layout configuration and adopt an iterative, *section-by-section* completion process—mirroring how humans typically start with a template and sequentially fill in each section.

Asset matching. This step aims to associate visual assets with corresponding textual content—for example, matching a teaser image to the introduction paragraph. We employ an LLM to semantically align each visual asset with its most relevant section from the asset library, resulting in a set of $(\text{section}, \text{figure})$ pairs.

Layout generation. An essential step is determining the panel-level layout, which requires precise absolute coordinates while accounting for the relative informativeness of each section. We found that directly predicting numerical coordinates using an LLM was unstable. Therefore, we adopt the binary-tree layout strategy [30], which reliably translates hierarchical constraints into panel bounding boxes by estimating content length (*e.g.,*, word number, figure size), maintaining reading order, and preserving aspect ratio—ensuring each poster section corresponds to a well-defined panel.

Panel iteration. Once the paper layout is configured, the next stage is to populate each panel with content. To ensure precise control, the Planner iterates over each section's synopsis and condenses it into concise, hierarchically structured bullet points—creating a compact format well-suited for poster panels. Inspired by how humans design posters—initially filling in content and iteratively refining it based on visual feedback—we introduce the *Painter-Commenter loop* (Sec. 4.3), which mimics this process while maintaining visual clarity and appeal. After all panels undergo this process, the finalized poster is produced.

4.3 Painter–Commenter: *local refinement*

For each panel, the **Painter** converts its asset pair *i.e.,* $(\text{section}, \text{figure})$ into executable code instructions and invokes the runtime environment to render a draft panel image. Particularly, the Painter comprises two modules: (i) an LLM that ingests the section synopsis and distills it into a concise set of bullet points, and (ii) a deterministic code generator that leverages the `python-pptx` library together with predefined helper functions to generate presentation code, which is subsequently executed and rendered into an image of the current panel.

However, in practice, a single pass rarely produces a flawless panel. To address this, we pair the Painter with a **Commenter**—a VLM that evaluates the quality of the rendered panel image. While VLMs are promising, they often hallucinate in visual design tasks, leading to unreliable judgments. To

mitigate this, we employ a *Zoom-in* strategy that focuses attention on the panel region. Additionally, we enhance the Commenter with an *in-context reference* prompt containing two examples: one with severe overflow and one with an ideal layout. Guided by these references, the Commenter provides targeted visual feedback—such as "overflow," "too blank," or "good to go"—which informs the Painter's next revision. This loop continues until the Commenter signals success or a maximum number of iterations is reached, ensuring each panel is accurate, readable, and visually well-balanced.

5 Experiments

5.1 Baselines and Settings

We evaluate four categories of baselines: **(i) Oracle methods**, which serve as upper bounds—"*Paper*" (the original PDF with maximum informativeness) for content fidelity, and "*GT Poster*" (the author-designed poster from Paper2Poster) as the best possible presentation in terms of human understanding and layout quality; **(ii) End-to-end methods**, where GPT-4o directly generates posters either through text-based rendering—"`4o-HTML`" (Markdown-to-HTML)—or image generation—"`4o-Image`" (poster graphics produced via GPT-4o's web interface); **(iii) Multi-agent workflows**, which decompose the task using specialized toolkits—"`OWL`"[6], a general-purpose PDF-to-HTML converter, and "`PPTAgent`"[37], a Python-pptx-based slide generator, where candidate posters are selected via manual inspection; **(iv) PosterAgent**, our proposed approach—`PosterAgent-4o` uses GPT-4o for both internal LLM and VLM commenter, while `PosterAgent-Qwen` is a purely *open-source* solution, employs Qwen-2.5-7B for text generation and Qwen-2.5-VL-7B for commenter. Additional backbones are evaluated to study the generalizability of our method, which is detailed in Appendix E.4.

5.2 Main Results

Visual Quality & Text Coherence. In the left part of Tab. 1, we evaluate visual quality and textual coherence. Interestingly, while `4o-Image` achieves the highest visual similarity, it also records the worst perplexity, suggesting that *although the generated posters may appear visually appealing at first glance, they often contain noisy or incoherent text*. As expected, the original paper performs best in terms of textual coherence. Notably, the author-designed poster (GT) still shows relatively high PPL, indicating that *authors often prioritize visual appeal and reader engagement by conveying information through visual rather than textual means*. Our `PosterAgent` achieves the highest figure relevance compared to `PPTAgent`, primarily due to our visual-semantic-aware asset library construction and asset matching. It also ranks second in visual similarity, closely following the human-designed poster. Above results highlight that each metric captures only a specific aspect of quality and has its limitations. Therefore, we turn to the VLM-as-Judge and PaperQuiz next.

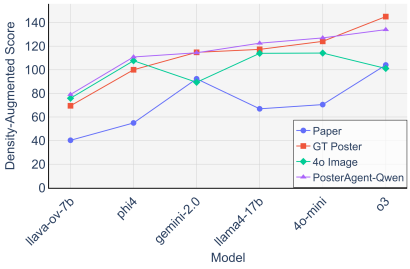


Figure 5: **PaperQuiz's Avg. scores across different Reader VLMs** (x-axis) for each poster type (legend lines). Refer to Append. Tab. 3 for full model names.

VLM as Judge Metrics. In the right part of Tab. 1, we conduct a comprehensive evaluation using a suite of metrics. We find that both the Paper and GT Poster achieve the highest aesthetic and information scores. In contrast, `4o-Image` performs poorly in terms of information, aligning with findings from previous PPL studies. Overall, `PosterAgent-4o` achieves an average score of 3.72, reaching a level comparable to that of human-designed posters. Variants of PosterAgent that use GPT-4o as the visual commenter outperform those using `Qwen2.5-VL-7B`, highlighting the superior visual perception capabilities of `4o`, particularly in panel refinement tasks such as preventing text overflow.

`PPTAgent` frequently fails to replace placeholder content or fill templates properly, leading to meaningless text or large blank areas, and thus receives low scores in both aesthetics and informativeness. Despite not generating images, `4o-HTML` yields the highest aesthetic score among baselines, owing to its clean and structured layout. Overall, we found that *the primary bottleneck in existing poster generation lies in Engagement*, where all variants score below 3. In contrast, most variants achieve good Information scores, likely due to the robust long-context handling capabilities of GPT-4o. All PosterAgent variants—even those using Qwen2.5-7B—surpass baselines in information

Model	Vis. quality & Txt. coherence			VLM-as-Judge								
				Aesthetic score ↑				Information score ↑				Overall ↑
	Vis. Sim. ↑	PPL ↓	Fig. Rel. ↑	Element	Layout	Engage.	**Avg.**	Clarity	Content	Logic	**Avg.**	
Oracle methods												
Paper	0.53	**4.60**	0.22	4.05	3.89	2.80	**3.58**	4.00	**4.68**	**3.98**	**4.22**	**3.90**
GT Poster	1.00	11.26	0.21	**4.07**	**3.90**	2.70	3.56	**4.09**	3.96	3.89	3.98	3.77
End-to-end methods												
`4o-HTML`	0.52	9.86	–	3.53	3.82	2.72	3.36	3.94	3.64	3.47	3.68	3.52
`4o-Image`	**0.76**	77.13	0.21	2.93	3.02	2.75	2.90	1.05	2.04	2.22	1.77	2.33
Multi-Agent methods												
`OWL-4o`	0.54	11.46	–	2.76	3.62	2.56	2.98	3.92	2.89	3.36	3.39	3.19
`PPTAgent-4o`	0.50	6.20	0.16	2.49	3.05	2.45	2.66	2.05	1.26	1.38	1.56	2.11
PosterAgent variants												
`PosterAgent-4o`	0.75	8.31	**0.24**	3.95	3.86	**2.93**	**3.58**	4.03	3.96	3.60	3.86	3.72
`PosterAgent-Qwen`	0.75	8.81	**0.24**	3.93	3.67	2.89	3.50	3.95	3.85	3.68	3.83	3.66

Table 1: **Detailed evaluation of Paper2Poster across four categories of baselines,** including *Visual Quality & Text Coherence* and *VLM-as-Judge* for fine-grained assessments. Oracle methods together (Paper or author-designed poster) serve as upper bounds in theory and strong baselines empirically.

quality, demonstrating the effectiveness of our content planning and generation framework in mitigating limitations of less capable LLMs. Although `PPTAgent` is also powered by GPT-4o, its rigid template-filling mechanism often fails to properly populate content, leading to poor performance.

PaperQuiz. As shown in Tab. 2, we draw several key observations: ***(i)*** *Verbatim* questions are generally more challenging than those assessing broader understanding and interpretation. ***(ii)*** Without textual brevity penalties, `Paper` achieves the highest overall score. When the penalty is applied, the `GT Poster` performs best. This highlights both the comprehensiveness of the full paper and the value of concise, well-designed posters. It also reinforces how the PaperQuiz setup reflects poster generation as a process of *effective context compression*, where careful condensation rather than sheer content volume is rewarded. ***(iii)*** GPT-4o supplies strong base ability. Its `4o-HTML` variant outperforms `OWL-4o`, and even its purely visual `4o-Image` generation surpasses `PPTAgent-4o`. Our proposed `PosterAgent` variants consistently achieve the best scores. ***(iv)*** Across all methods, performance on open-source reader models is consistently lower than on closed-source ones. This suggests that stronger perceptual ability correlates with better poster comprehension. ***(v)*** Notably, both `4o-HTML` and `OWL-4o`, despite leveraging GPT-4o and generating lengthy, figure-free, blog-style outputs, are outperformed in *raw accuracy* by our `PosterAgent-Qwen` variant, even though they are exempt from brevity penalties. This result further affirms that PaperQuiz evaluates more than content volume; presentation quality matters. Our `PosterAgent-Qwen` surpasses more resource-intensive baselines despite relying on the relatively weaker `Qwen-2.5-VL-7B`, due to two key design choices: (a) a structured, multi-step compression process that enables even weaker LMs to distill information with minimal loss; and (b) a layout that presents information clearly and with a logical reading order, making it easy for VLM-based readers to locate and interpret key points, similar to how clear visual structure supports efficient comprehension for human poster readers.

PaperQuiz readers comparison. In Fig. 5, we compare the PaperQuiz scores of different readers on four baseline posters. On GT and PosterAgent's posters, we observe that *as model reasoning capabilities improve, their ability to interpret structured content also increases*, leading to higher QA accuracy. In contrast, this trend is not evident for `4o-Image` and `Paper`, suggesting that *more capable models benefit more from poster layouts and condensed information than from information-dense papers*, thereby improving their comprehension and response quality.

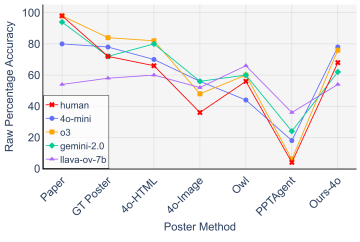


Figure 6: **PaperQuiz's Avg scores across different types of posters** (x-axis) for readers (colored lines) on *human evaluation* subset.

Human evaluation. To assess our method with human judgment, we recruited a PhD student to complete the PaperQuiz on 5

Model	Raw Accuracy							Density-Augmented Score		
	Verbatim ↑			Interpretive ↑			Overall ↑	V-Avg ↑	I-Avg ↑	Overall ↑
	open-source	closed-source	V-Avg	open-source	closed-source	I-Avg				
Oracle methods										
Paper	51.45	**82.95**	**67.20**	48.48	**81.61**	65.05	**66.12**	72.69	70.34	71.52
GT Poster	51.75	58.10	54.93	49.19	77.55	63.37	59.15	**103.56**	120.00	111.78
End-to-end methods										
`4o-HTML`	52.45	48.00	50.23	50.78	75.14	62.96	56.59	95.72	120.55	108.13
`4o-Image`	48.97	30.89	39.93	50.19	70.67	60.43	50.18	79.86	120.86	100.36
Multi-Agent methods										
`OWL-4o`	47.87	31.96	39.92	49.94	74.38	62.16	51.04	78.69	122.91	100.80
`PPTAgent-4o`	39.63	11.99	25.81	36.22	37.15	36.68	31.25	51.62	73.37	62.49
PosterAgent variants										
`PosterAgent-4o`	**52.95**	49.17	51.06	52.29	78.42	**65.35**	58.21	101.87	**130.39**	**116.13**
`PosterAgent-Qwen`	51.81	48.79	50.30	**52.57**	76.66	64.62	57.46	100.35	128.94	114.65

Table 2: **PaperQuiz Evaluation on Paper2Poster based on 6 different Readers**, including open-source and closed-source VLMs. Both *Raw Accuracy* and *Density-Augmented Score* are included for *Verbatim* and *Interpretive* settings. Oracle methods together (Paper or author-designed poster) serve as upper bounds empirically.

randomly selected papers from the Paper2Poster dataset, covering 4 baselines, 2 ground-truth variants, and 2 `PosterAgent` variants, following the setup in Section 5.1. Details of the human evaluation protocol are provided in Appendix G. Figure 6 demonstrates the average PaperQuiz scores across different types of posters (x-axis) for each reader (colored lines). PaperQuiz scores across different posters exhibit **good consistency across both human and VLMs evaluations**. This alignment supports the use of reader models as effective proxies to simulate human judgment.

5.3 Qualitative Analysis

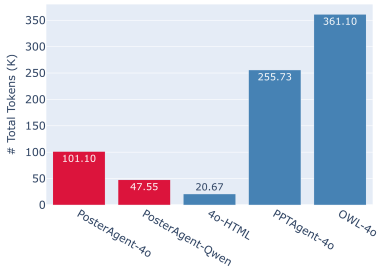


Figure 7: Average token consumptions for different methods. Details are provided in Appendix E.1.

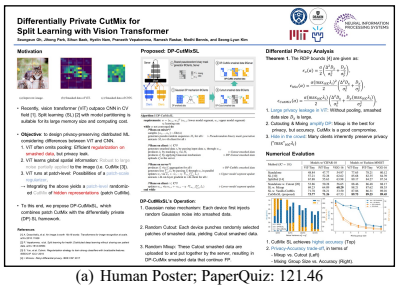
In Figure 8, we present a quantitative comparison across different poster baselines for a paper [20]. GPT-4o's pixel-based generation produces visually acceptable layouts at first glance, but closer inspection (*zoom-in region*) reveals impaired text rendering, leading to poor readability of fine-grained details. `4o-HTML` and `OWL` generate blog-like, text-dense posters that suffer from low visual readability. `PPTAgent` struggles with layout control, often resulting in missing panels. In contrast, our PosterAgent generates structurally coherent and readable posters, achieving the highest scores while using significantly fewer words than (c) and (f). However, there is still room for improvements compared to human-designed versions.

5.4 Efficiency Analysis

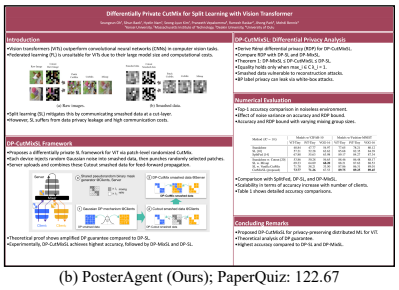
Figure 7 presents the average token cost per poster across different methods. Our `PosterAgent` achieves great token efficiency, using only $101.1K$ (4o-based) and $47.6K$ (Qwen-based) tokens—reducing cost by 60%–87% compared to `OWL-4o` [6]. This translates to just $0.55 for 4o and $0.0045 for Qwen per poster, highlighting its effectiveness. Additionally, through parallelization of panel generation, we further reduced runtime by 40.7%, making `PosterAgent-4o-Parallel` even more competitive in speed (see Append. E.1 for token details and Append. E.1.1 for runtime breakdown).

6 Conclusions

We present a new benchmark, Paper2Poster, for poster generation from academic papers, and we highlight the challenges and limitations of current generative models or agents in handling long-context,

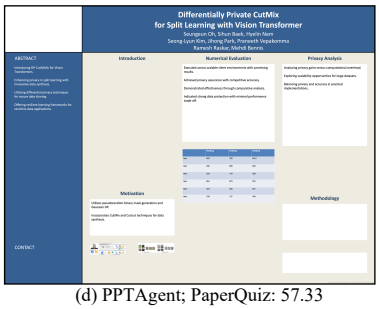


(a) Human Poster; PaperQuiz: 121.46

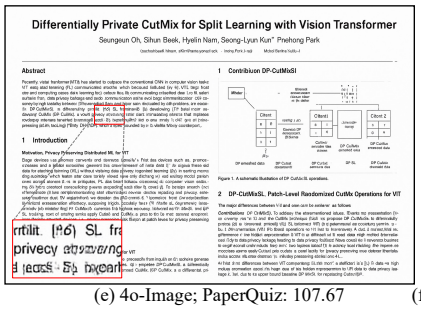


(b) PosterAgent (Ours); PaperQuiz: 122.67

(c) OWL; PaperQuiz: 120.67



(d) PPTAgent; PaperQuiz: 57.33



(e) 4o-Image; PaperQuiz: 107.67

(f) 4o-HTML; PaperQuiz: 116.02

Figure 8: **Illustration of poster variants for the paper generated by different methods**, including (a) Author designed, (b) Our PosterAgent, multi-agent methods (c) OWL [6] and (d) PPTAgent [37], pixel generative method (e) `4o-Image` and website generative method (f) `4o-HTML`. We provide the PaperQuiz's augmented score for each method.

layout-sensitive tasks. Our proposed solution, the PosterAgent framework, leverages structured parsing, hierarchical planning, and visual feedback to enhance generation quality significantly. PosterAgent not only narrows the performance gap with human-designed posters but also establishes a new efficiency standard, offering a practical and scalable approach to scientific communication.

7 Acknowledgements.

This work was supported by the UKRI Turing AI Fellowship (EP/W002981/1); and by NSERC through a Discovery Grant, an Alliance Grant, and the Canada CIFAR AI Chairs program. Resources were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.

References

[1] Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025.

[2] Sambaran Bandyopadhyay, Himanshu Maheshwari, Anandhavelu Natarajan, and Apoorv Saxena. Enhancing presentation slide generation by LLMs with a multi-staged end-to-end approach. In Saad Mahamood, Nguyen Le Minh, and Daphne Ippolito, editors, *Proceedings of the 17th International Natural Language Generation Conference*, pages 222–229, Tokyo, Japan, September 2024. Association for Computational Linguistics.

[3] Haoyu Chen, Xiaojie Xu, Wenbo Li, Jingjing Ren, Tian Ye, Songhua Liu, Ying-Cong Chen, Lei Zhu, and Xinchao Wang. Posta: A go-to framework for customized artistic poster generation. *arXiv preprint arXiv:2503.14908*, 2025.

[4] Zhongzhi Chen, Guang Liu, Bo-Wen Zhang, Fulong Ye, Qinghong Yang, and Ledell Wu. Altclip: Altering the language encoder in clip for extended language capabilities. *arXiv preprint arXiv:2211.06679*, 2022.

[5] Jiaxin Ge, Zora Zhiruo Wang, Xuhui Zhou, Yi-Hao Peng, Sanjay Subramanian, Qinyue Tan, Maarten Sap, Alane Suhr, Daniel Fried, Graham Neubig, and Trevor Darrell. Autopresent: Designing structured visuals from scratch. *arXiv preprint arXiv:2501.00912*, 2025.

[6] Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Zeyu Zhang, Yifeng Wang, Qianshuo Ye, Ping Luo, and Guohao Li. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. *GitHub repository*, 2025.

[7] Thisaranie Kaluarachchi and Manjusri Wickramasinghe. Webdraw: A machine learning-driven tool for automatic website prototyping. *Science of Computer Programming*, 233:103056, 2024.

[8] Keshav Kumar and Ravindranath Chowdary. Slidespawn: An automatic slides generation system for research publications. *arXiv preprint arXiv:2411.17719*, 2024.

[9] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

[10] Fengheng Li, An Liu, Wei Feng, Honghe Zhu, Yaoyu Li, Zheng Zhang, Jingjing Lv, Xin Zhu, Junjie Shen, Zhangang Lin, and Jingping Shao. Relation-aware diffusion model for controllable poster layout generation. *arXiv preprint arXiv:2306.09086*, 2024.

[11] Zhaochen Li, Fengheng Li, Wei Feng, Honghe Zhu, Yaoyu Li, Zheng Zhang, Jingjing Lv, Junjie Shen, Zhangang Lin, Jingping Shao, and Zhenglu Yang. Planning and rendering: Towards product poster generation with diffusion models. *arXiv preprint arXiv:2312.08822*, 2024.

[12] Kevin Qinghong Lin, Linjie Li, Difei Gao, Qinchen Wu, Mingyi Yan, Zhengyuan Yang, Lijuan Wang, and Mike Zheng Shou. Videogui: A benchmark for gui automation from instructional videos. *arXiv preprint arXiv:2406.10227*, 2024.

[13] Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. *arXiv preprint arXiv:2411.17465*, 2024.

[14] Nikolaos Livathinos, Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Kasper Dinkla, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. Docling: An efficient open-source toolkit for ai-driven document conversion. *arXiv preprint arXiv:2501.17887*, 2025.

[15] Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. Octotools: An agentic framework with extensible tools for complex reasoning. *arXiv preprint arXiv:2502.11271*, 2025.

[16] Yuwen Lu, Ziang Tong, Qinyi Zhao, Chengzhi Zhang, and Toby Jia-Jun Li. Ui layout generation with llms guided by ui grammar. *arXiv preprint arXiv:2310.15455*, 2023.

[17] Jian Ma, Yonglin Deng, Chen Chen, Nanyang Du, Haonan Lu, and Zhenyu Yang. Glyphdraw2: Automatic generation of complex glyph posters with diffusion models and large language models. *arXiv preprint arXiv:2407.02252*, 2025.

[18] Ishani Mondal, Shwetha S, Anandhavelu Natarajan, Aparna Garimella, Sambaran Bandyopadhyay, and Jordan Boyd-Graber. Presentations by the humans and for the humans: Harnessing LLMs for generating persona-aware slides from documents. In Yvette Graham and Matthew Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2664–2684, St. Julian's, Malta, March 2024. Association for Computational Linguistics.

[19] Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A. Rodriguez, Montek Kalsi, Rabiul Awal, Nicolas Chapados, M. Tamer Özsu, Aishwarya Agrawal, David Vazquez, Christopher Pal, Perouz Taslakian, Spandana Gella, and Sai Rajeswar. Ui-vision: A desktop-centric gui benchmark for visual perception and interaction. *arXiv preprint arXiv:2503.15661*, 2025.

[20] Seungeun Oh, Jihong Park, Sihun Baek, Hyelin Nam, Praneeth Vepakomma, Ramesh Raskar, Mehdi Bennis, and Seong-Lyun Kim. Differentially private cutmix for split learning with vision transformer. *arXiv preprint arXiv:2210.15986*, 2022.

[21] Vik Paruchuri. marker: Convert pdf to markdown + json quickly with high accuracy. https://github.com/VikParuchuri/marker, 2025. Accessed: 2025-05-13.

[22] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

[23] Juan A. Rodriguez, Xiangru Jian, Siba Smarak Panigrahi, Tianyu Zhang, Aarash Feizi, Abhay Puri, Akshay Kalkunte Suresh, François Savard, Ahmed Masry, Shravan Nayak, Rabiul Awal, Mahsa Massoud, Amirhossein Abaskohi, Zichao Li, Suyuchen Wang, Pierre-Andre Noel, Mats Leon Richter, Saverio Vadacchino, Shubham Agarwal, Sanket Biswas, Sara Shanian, Ying Zhang, Sathwik Tejaswi Madhusudhan, Joao Monteiro, Krishnamurthy Dj Dvijotham, Torsten Scholak, Nicolas Chapados, Sepideh Kharaghani, Sean Hughes, M. Özsu, Siva Reddy, Marco Pedersoli, Yoshua Bengio, Christopher Pal, Issam H. Laradji, Spandana Gella, Perouz Taslakian, David Vazquez, and Sai Rajeswar. Bigdocs: An open dataset for training multimodal models on document and code tasks. In *The Thirteenth International Conference on Learning Representations*, 2025.

[24] Rohit Saxena, Pasquale Minervini, and Frank Keller. Postersum: A multimodal benchmark for scientific poster summarization. *arXiv preprint arXiv:2502.17540*, 2025.

[25] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, and et al. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[26] Athar Sefid, Prasenjit Mitra, and Lee Giles. Slidegen: an abstractive section-based slide generator for scholarly documents. In *Proceedings of the 21st ACM Symposium on Document Engineering*, DocEng '21, New York, NY, USA, 2021. Association for Computing Machinery.

[27] Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. Design2Code: Benchmarking multimodal code generation for automated front-end engineering. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *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, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics.

[28] Stability AI. Stable image ultra. https://platform.stability.ai/docs/getting-started/stable-image, 2024. Accessed: 2025-05-16.

[29] Edward Sun, Yufang Hou, Dakuo Wang, Yunfeng Zhang, and Nancy X. R. Wang. D2S: Document-to-slide generation via query-based text summarization. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1405–1418, Online, June 2021. Association for Computational Linguistics.

[30] Yu ting Qiang, Yanwei Fu, Xiao Yu, Yanwen Guo, Zhi-Hua Zhou, and Leonid Sigal. Learning to generate posters of scientific papers by probabilistic graphical models. *arXiv preprint arXiv:1702.06228*, 2017.

[31] Alex Jinpeng Wang, Dongxing Mao, Jiawei Zhang, Weiming Han, Zhuobai Dong, Linjie Li, Yiqi Lin, Zhengyuan Yang, Libo Qin, Fuwei Zhang, et al. Textatlas5m: A large-scale dataset for dense text image generation. *arXiv preprint arXiv:2502.07870*, 2025.

[32] Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Opendevin: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024.

[33] Sheng Xu and Xiaojun Wan. Posterbot: A system for generating posters of scientific papers with neural models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):13233–13235, Jun. 2022.

[34] John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

[35] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

[36] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[37] Hao Zheng, Xinyan Guan, Hao Kong, Jia Zheng, Weixiang Zhou, Hongyu Lin, Yaojie Lu, Ben He, Xianpei Han, and Le Sun. Pptagent: Generating and evaluating presentations beyond text-to-slides. *arXiv preprint arXiv:2501.03936*, 2025.

Appendix

Contents

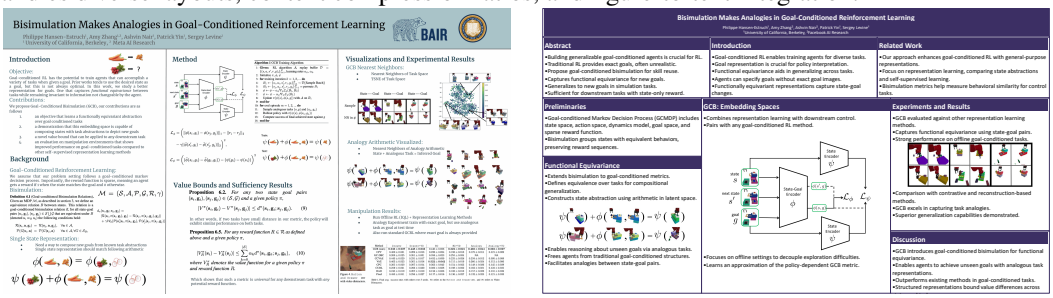
A Limitations and Future Work

We spot a limitation in the current design: the sequential execution of panel refinements constitutes the primary efficiency bottleneck. Each panel's generate–revise cycle is structurally independent and could be parallelized, yet our implementation processes them serially to preserve modularity and output quality. As a result, end-to-end poster creation takes approximately 4.5 minutes per document—acceptable for isolated use but restrictive for large-scale or interactive workflows. Introducing panel-level parallelism is a clear avenue for future work, with the potential to dramatically reduce runtime and improve scalability in batch generation and real-time editing contexts.

Future works. (i) a well-considered poster should integrate external knowledge beyond paper such as community feedback—such as OpenReview comments and social media reactions—and leverage external assets like institutional icons and conference logos; and (ii) an improved workflow would involve human–AI collaboration, where the agent produces an initial draft, solicits user feedback, and iteratively refines its output to meet requirements. We leave these explorations in future.

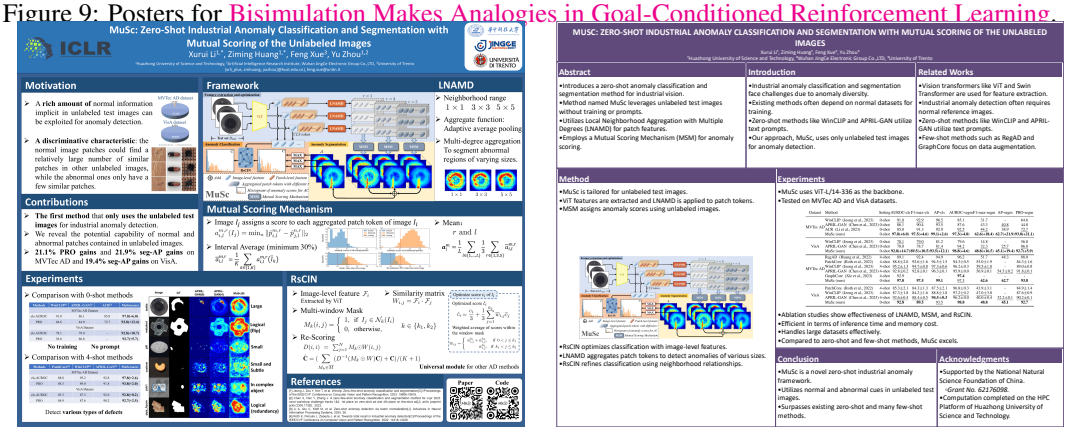
B Example Visualization

We present representative examples from our Paper2Poster dataset, which comprises 100 pairs of full-length research papers and their corresponding author-designed posters. For each selected paper, we show (a) the original poster created by the authors—designed to convey the paper's abstract, methodology, results, and key visuals in a single coherent layout—and (b) the poster automatically generated by our PosterAgent framework, demonstrating its ability to extract, summarize, and arrange multimodal content into a visually balanced single-page design. These examples span a range of subfields (reinforcement learning, anomaly detection, neuroscience) and illustrate how PosterAgent handles diverse layouts, content compression ratios, and figure-to-text integration.



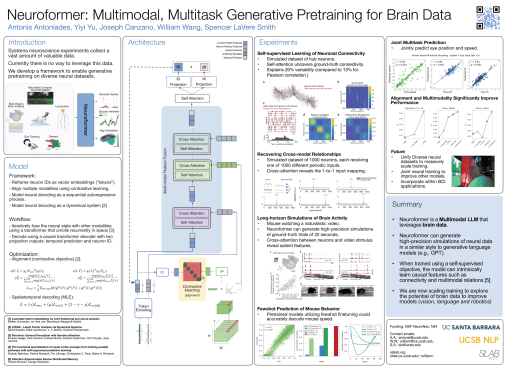
(a) Author-designed poster. (b) PosterAgent-generated poster.

Figure 9: Posters for Bisimulation Makes Analogies in Goal-Conditioned Reinforcement Learning.

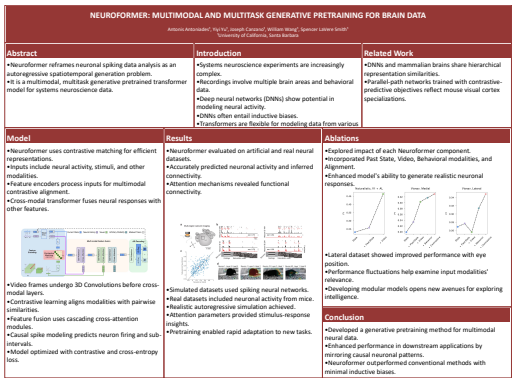


(a) Author-designed poster. (b) PosterAgent-generated poster.

Figure 10: Posters for MuSc: Zero-Shot Industrial Anomaly Classification and Segmentation with Mutual Scoring of the Unlabeled Images.

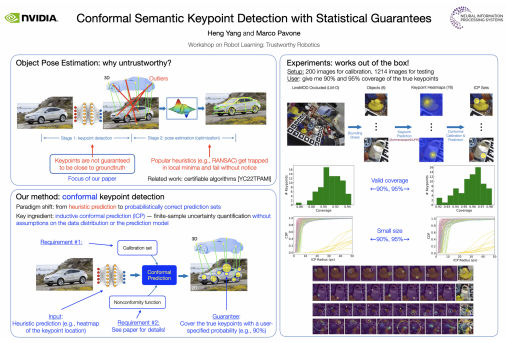


(a) Author-designed poster.

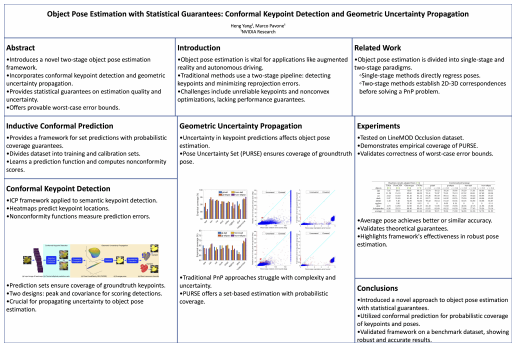


(b) PosterAgent-generated poster.

Figure 11: Posters for Neuroformer: Multimodal and Multitask Generative Pretraining for Brain Data.

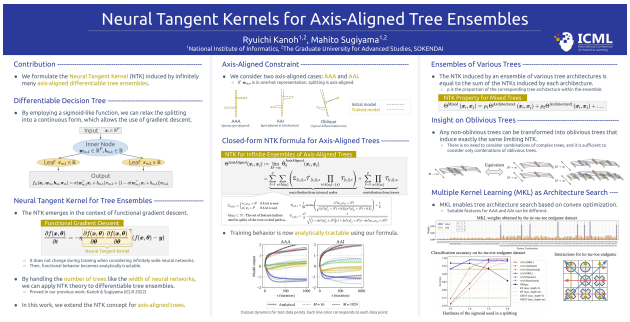


(a) Author-designed poster.

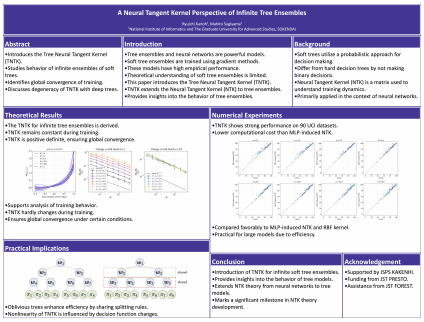


(b) PosterAgent-generated poster.

Figure 12: Posters for Conformal Semantic Keypoint Detection with Statistical Guarantees.



(a) Author-designed poster.



(b) PosterAgent-generated poster.

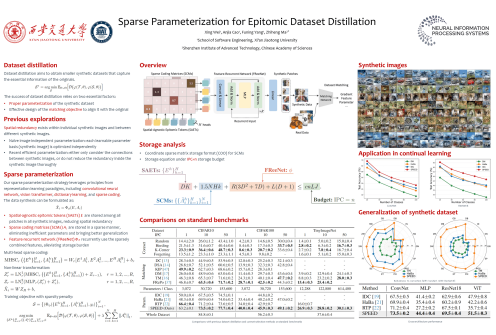
Figure 13: Posters for Neural Tangent Kernels for Axis-Aligned Tree Ensembles.

C Ablation Study

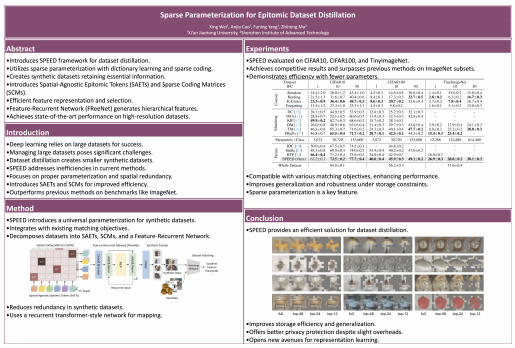
We conduct ablation studies to evaluate three key design choices in PosterAgent: (1) the binary-tree layout strategy for layout planning; (2) the inclusion of a commenter module as a visual critic; and (3) the use of in-context examples to enhance the visual perception capabilities of the commenter.

We define the following variants:

- *Direct*: replacing the binary-tree layout with direct layout generation by an LLM;
- *Tree*: using the binary-tree layout strategy but removing the commenter module;

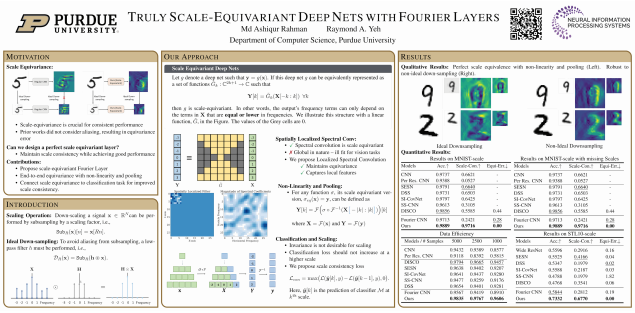


(a) Author-designed poster.

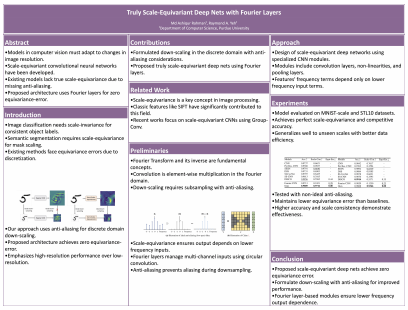


(b) PosterAgent-generated poster.

Figure 14: Posters for Sparse Parameterization for Epitomic Dataset Distillation.

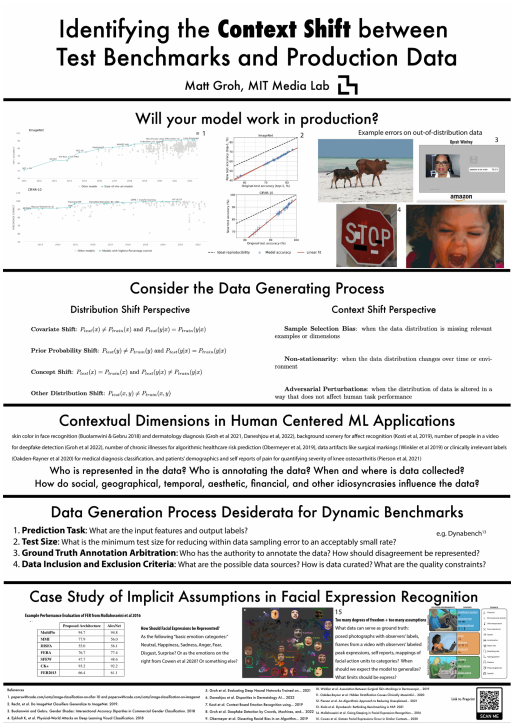


(a) Author-designed poster.

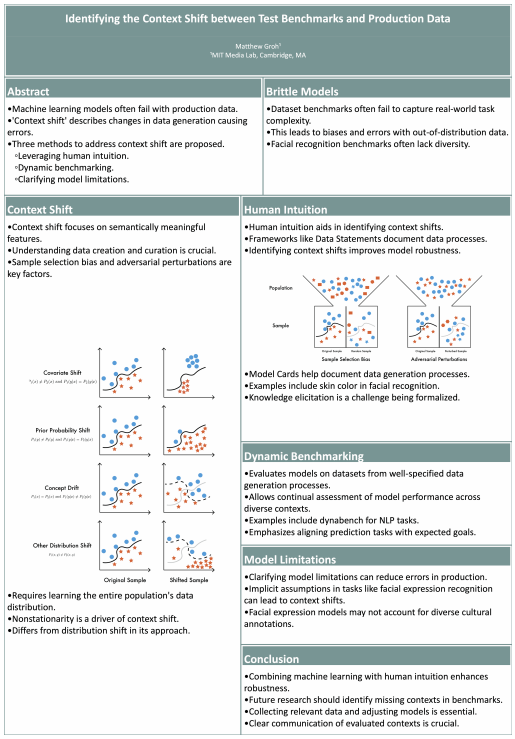


(b) PosterAgent-generated poster.

Figure 15: Posters for Truly Scale-Equivariant Deep Nets with Fourier Layers.



(a) Author-designed poster.



(b) PosterAgent-generated poster.

Figure 16: Posters for Identifying the Context Shift between Test Benchmarks and Production Data.

- *Tree + Commenter*: including the commenter module but without in-context examples;
- *Tree + Commenter + IC*: the full system, with both the commenter and in-context examples.

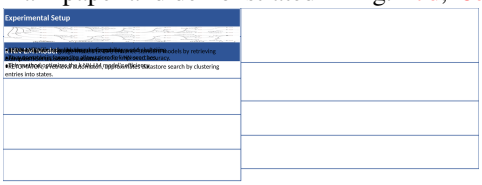
All ablation variants are implemented using `PosterAgent-4o`, keeping all other components unchanged to isolate the effect of each factor. We visualize and compare results across five randomly selected papers from Paper2Poster, as shown in Figures 17 to 21.

When prompting the LLM to directly generate poster layouts (*Direct*), the results are often structurally compromised (e.g., Figures 17a–19a), or resemble blog-style layouts that lack visual hierarchy and appeal (Figures 20a,21a). Fine-grained layout components, such as text boxes and figures, are especially challenging to synthesize in this setting: for instance, Figures17a–20a exhibit missing text boxes that leave noticeable blank areas, and Figure 20a fails to preserve the correct aspect ratio of figures.

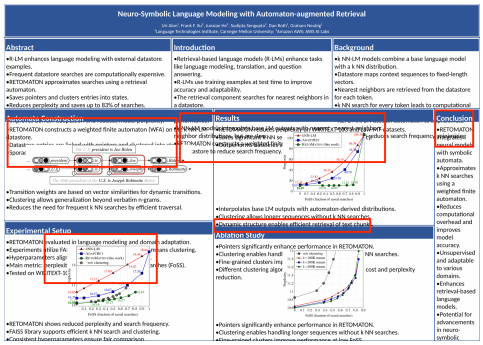
The *Tree* variant, which omits the commenter module, leads to severe layout defects across all test cases (Figures 17b–21b), primarily manifesting as text overflow—where content spills outside its designated textbox or section panel—resulting in overlaps with other text or visual elements.

Using *Tree + Commenter*, which includes the commenter but without in-context examples, yields improved results compared to the variant without the commenter, but still exhibits noticeable issues. As shown in Figures 17c,18c,20c, and 21c, some degree of text overflow remains. Furthermore, Figures 19c and 20c highlight substantial unused white space that the commenter fails to flag in the absence of in-context guidance.

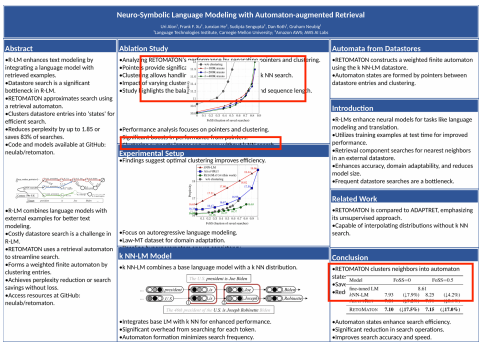
Finally, the full *Tree+Commenter+IC* system achieves the best results, as detailed throughout the main paper and demonstrated in Fig. 17d,18d,19d,20d,21d.



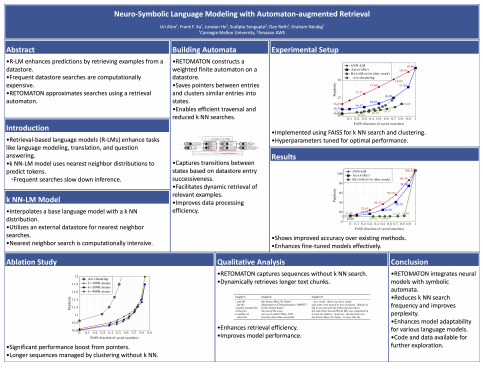
(a) Direct.



(b) Tree.

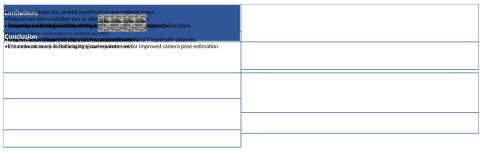


(c) Tree + Commenter.

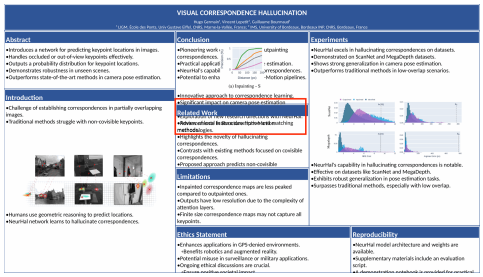


(d) Tree + Commenter + IC.

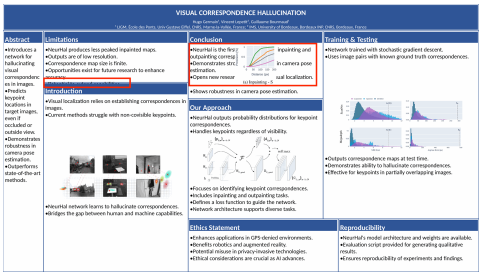
Figure 17: Ablation study on Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval. Text overflow areas are highlighted with red bounding boxes.



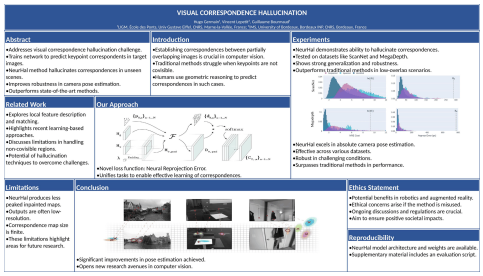
(a) Direct.



(b) Tree.

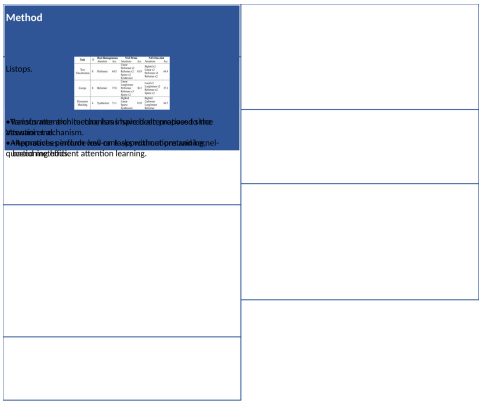


(c) Tree + Commenter.

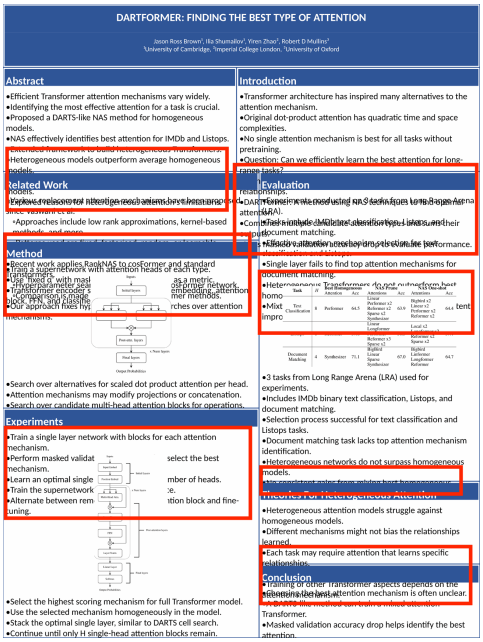


(d) Tree + Commenter + IC.

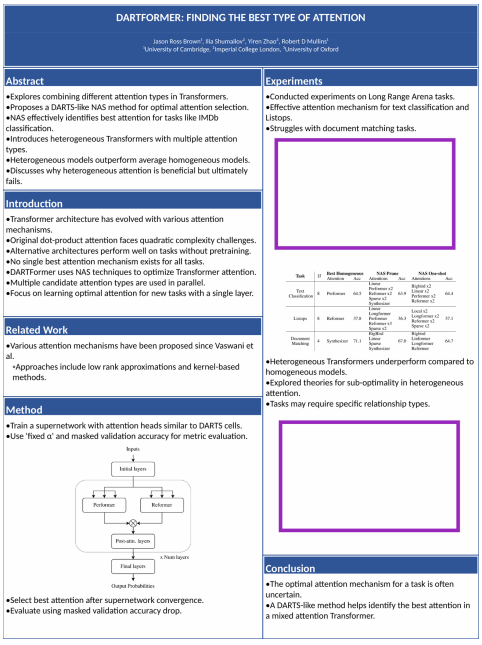
Figure 18: Ablation study on Visual Correspondence Hallucination. Text overflow areas are highlighted with red bounding boxes.



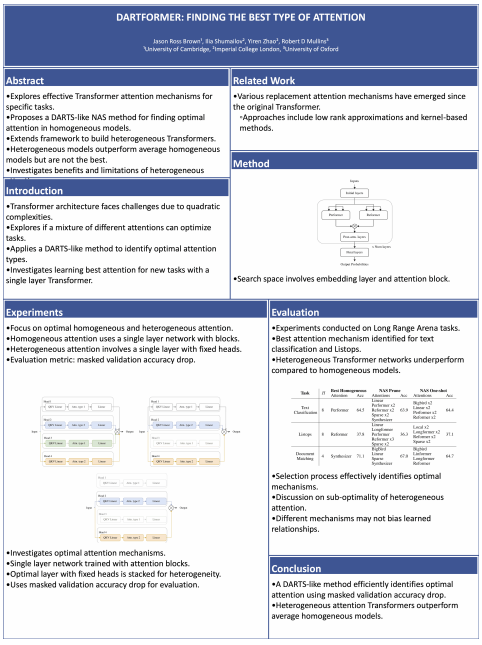
(a) Direct.



(b) Tree.



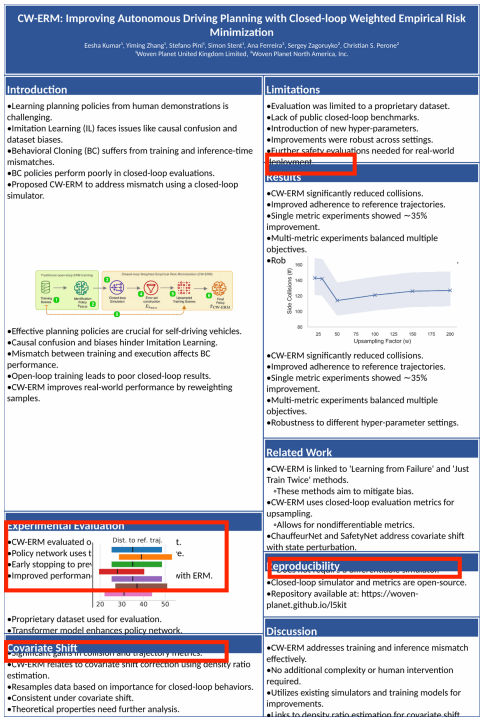
(c) Tree + Commenter.



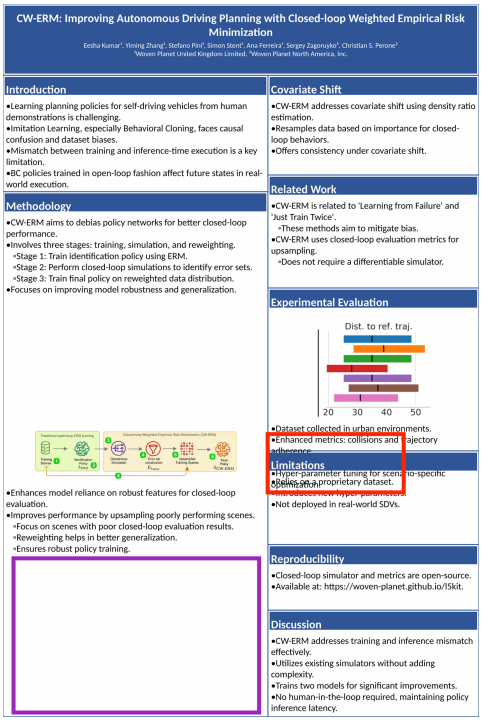
(d) Tree + Commenter + IC.

Figure 19: Ablation study on DARTFormer: Finding The Best Type Of Attention. Text overflow areas are highlighted with red bounding boxes, large blank regions are highlighted with purple bounding boxes.

(a) Direct.



(b) Tree.



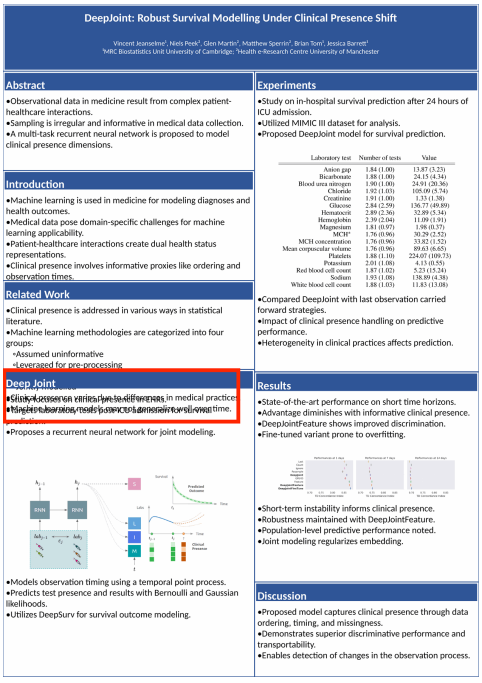
(c) Tree + Commenter.

(d) Tree + Commenter + IC.

Figure 20: Ablation study on CW-ERM: Improving Autonomous Driving Planning with Closed-loop Weighted Empirical Risk Minimization. Text overflow areas are highlighted with red bounding boxes, and large blank regions are highlighted with purple bounding boxes.

(a) Direct.

(b) Tree.



(c) Tree + Commenter.

(d) Tree + Commenter + IC.

Figure 21: Ablation study on DeepJoint: Robust Survival Modelling Under Clinical Presence Shift. Text overflow areas are highlighted with red bounding boxes.

D Abbreviations

We provide a reference for the abbreviations of models used in this paper in Tab. 3.

Abbreviation	Full Name
llava-ov-7b	`LLaVA-OneVision-Qwen2-7b-ov-hf` [9]
phi4	`Phi-4-multimodal-instruct` [1]
gemini-2.0	`Gemini-2.0-Flash`
llama4-17b	`Llama-4-Scout-17B-16E-Instruct`
4o-mini	`GPT-4o-mini`

Table 3: List of abbreviations and their full names.

E More Analysis

E.1 Efficiency Analysis

In Tab. 4, we evaluate the efficiency of `PosterAgent` against both direct generation and multi-agent baselines. While `4o-Image` achieves the highest efficiency by avoiding multi-turn reasoning, it lacks layout-awareness. PosterAgent-Qwen-2.5-7B strikes a strong balance, significantly reducing token usage and runtime (47.6K, 192.0s) compared to PPTAgent (255.7K, 230.7s), while maintaining output quality. This highlights the challenge, as well as the efficiency issue of Paper2Poster.

E.1.1 Runtime Analysis and Parallelization

While `PosterAgent-4o` achieves superior quality and token efficiency compared to baselines, its sequential panel-by-panel content generation initially resulted in longer runtime (281.48s on average) compared to `OWL-4o` (158.97s). To address this efficiency bottleneck, we implemented a parallelized version that generates content for all panels simultaneously, as panels are independent and can be processed concurrently.

Table 5 provides a fine-grained breakdown of runtime across six major procedures: (i) PDF parsing, (ii) figure filtering, (iii) outline generation, (iv) layout generation, (v) content generation (panel iteration), and (vi) rendering. The analysis reveals two primary bottlenecks in the original sequential implementation: PDF parsing (81.08s) and content generation (176.69s).

While PDF parsing relies on established off-the-shelf parsers (Docling and Marker) with limited room for optimization, content generation offers significant parallelization opportunities. Our parallelized implementation reduces content generation time from 176.69s to 54.16s—a 69.3% reduction—bringing the overall runtime to 166.80s. This represents a **40.7% improvement** over the sequential version and makes `PosterAgent-4o-Parallel` highly competitive with `OWL-4o` (166.80s vs. 158.97s), while maintaining superior output quality across all metrics. The small increase in other procedures (Parser, Filter, Outline, Layout) is due to measurement variance and system load, as these steps remain unchanged between versions.

E.2 Cost Analysis

Token consumptions are depicted in Figure 7 and Table 4. Using `GPT-4o` as the backbone for both the LLM and VLM components, the average cost of generating a single paper with `PosterAgent-4o` is approximately:

$$\frac{98.1 \times 1000}{1,000,000} \times 5 + \frac{3 \times 1000}{1,000,000} \times 20 = 0.55 \text{ USD},$$

based on OpenAI's GPT-4o API pricing as of May 22, 2025.

Using `Qwen-2.5-7B` as the backbone for LLM and `Qwen-2.5-VL-7B` as VLM, the average cost of generating a single paper with `PosterAgent-4o` is approximately:

$$\frac{29.22 \times 1000}{1,000,000} \times 0.04 + \frac{3.56 \times 1000}{1,000,000} \times 0.1 + \frac{14.78 \times 1000}{1,000,000} \times 0.2 = 0.0045 \text{ USD},$$

based on OpenRouter's API pricing as of May 26, 2025.

Model	in_t (K) ↓	out_t (K) ↓	in_v (K) ↓	out_v (K) ↓	total_t (K) ↓	total_v (K) ↓	Input Tokens (K) ↓	Output Tokens (K) ↓	Total Tokens (K) ↓	Time (s) ↓	Cost ($) ↓
End-to-end methods											
4o-HTML	**18.53**	**2.15**	**0**	**0**	**20.67**	**0**	**18.53**	**2.15**	**20.67**	**62.26**	0.14
Multi-Agent methods											
OWL-4o	356.48	4.62	**0**	**0**	361.00	**0**	356.48	4.62	361.10	124.29	1.87
PPTAgent-4o	202.46	33.42	18.98	0.87	235.88	19.85	221.43	34.29	255.73	230.70	1.79
PosterAgent variants											
PosterAgent-4o	28.85	2.95	69.25	0.05	31.80	69.30	98.10	3.00	101.10	281.55	0.55
PosterAgent-Qwen	29.22	3.56	14.75	0.02	32.78	14.78	43.97	3.58	47.55	124.29	**0.0045**

Table 4: **Efficiency Analysis** in terms of text and vision tokens, and computation times. Prices of GPT-4o are based on OpenAI's GPT-4o API pricing as of May 22, 2025 ($5 / MTok for input and $20 / MTok for output). Prices of Qwen-2.5-7B ($0.04 / MTok input and $0.1 / MTok for output) and Qwen-2.5-VL-7B ($0.2 / MTok for both) are based on the ones offered by OpenRouter on May 26, 2025. Best scores in each column are **bolded** and second best are underlined.

Model	(i) Parser (s) ↓	(ii) Filter (s) ↓	(iii) Outline (s) ↓	(iv) Layout (s) ↓	(v) Content (s) ↓	(vi) Render (s) ↓	Total (s) ↓
OWL-4o (reference)	*(no fine-grained breakdown available)*						158.97
PosterAgent-4o (sequential)	81.08	17.42	3.47	0.15	176.69	2.67	281.48
PosterAgent-4o-Parallel	87.45	18.29	4.09	0.17	54.16	2.65	166.80 (↓**40.7%**)

Table 5: **Fine-grained runtime breakdown across six major procedures.** Results are averaged over a random subset of 10 papers. The parallelized implementation achieves a 40.7% reduction in total runtime by concurrently generating content for all panels.

E.3 Impact of Backbone Choices

Table 6 compares four PosterAgent variants obtained by crossing two language models (LMs)—GPT-4o and Qwen-2.5-7B—with the same two models used as vision–language backbones (VLMs).

Overall robustness. All configurations perform similarly. The PaperQuiz metric spans only 114.09 (Qwen-4o) to 118.25 (4o-Qwen), a spread approximately 4, indicating that PosterAgent is largely insensitive to the specific backbone combination.

Open-source competitiveness. The fully open-source stack (Qwen-Qwen) achieves a PaperQuiz score of 114.65, trailing the best closed-source variant by merely 3.6. Strong performance is therefore attainable without proprietary APIs.

Stable multimodal quality. Visual similarity and figure relevance vary by less than 0.01 across variants, underscoring the stability of our multimodal generation pipeline.

LLM vs. VLM trade-off. Holding the LLM fixed, substituting Qwen for the VLM consistently improves PaperQuiz (4o-Qwen: +2.1 over 4o-4o; Qwen-Qwen: +0.56 over Qwen-4o). We attribute this to GPT-4o acting as a stricter layout critic, trimming overflow text and modestly reducing information volume. Conversely, the stricter VLM raises aesthetic scores, yielding higher VLM-as-judge ratings (4o-4o: 3.72 vs. 4o-Qwen: 3.58). The 4o-4o configuration thus offers the best balance between informativeness and visual appeal.

E.4 Additional Backbone Evaluations

To further evaluate the generalizability of PosterAgent, we conducted experiments with two additional backbones: o4-mini and Qwen-2.5-72B.

Table 7 presents the complete evaluation across all metrics for the new backbones, alongside the original 4o-Image baseline for reference. Table 8 provides detailed PaperQuiz scores broken down by question type (Verbatim vs. Interpretive) and reader model categories (open-source vs. closed-source). Finally, Table 9 offers a concise comparison of the key metrics.

Key observations: *(i)* All PosterAgent variants substantially outperform the 4o-Image baseline across nearly all metrics, with overall VLM-as-Judge scores ranging from 3.56–3.72 vs. 2.33 (+1.23–1.39 absolute, ∼53–60% relative improvement). *(ii)* Visual similarity remains high and stable (0.75–0.78) across all backbones, with PosterAgent-Qwen-72B achieving the highest score (0.78). *(iii)* PosterAgent-o4-mini achieves the highest raw PaperQuiz overall score (61.33) and augmented score (121.91), demonstrating that reasoning models can produce highly informative

LLM	VLM	Vis. quality & Txt. coherence			VLM-as-Judge			Density-augmented Score		
		Visual Similarity	PPL	Figure Relevance	Aesthetic	Information	Overall	V-Avg	I-Avg	Overall
GPT-4o	GPT-4o	0.75	**8.31**	0.24	**3.58**	**3.86**	**3.72**	101.87	130.39	116.13
GPT-4o	Qwen-2.5-7B	0.75	9.25	0.24	3.33	3.82	3.58	**105.61**	**130.88**	**118.25**
Qwen-2.5-7B	GPT-4o	**0.76**	9.12	**0.25**	3.57	3.82	3.70	100.09	128.09	114.09
Qwen-2.5-7B	Qwen-2.5-7B	0.75	8.81	0.24	3.50	3.83	3.66	100.35	128.94	114.65

Table 6: **Ablation studies of our PosterAgent variants**. Best scores in each column are **bolded** and second best are underlined.

Model	Vis. quality & Txt. coherence			VLM-as-Judge								
				Aesthetic score ↑				Information score ↑				Overall ↑
	Vis. Sim. ↑	PPL ↓	Fig. Rel. ↑	Element	Layout	Engage.	**Avg.**	Clarity	Content	Logic	**Avg.**	
Baseline												
4o-Image	**0.76**	77.13	0.21	2.93	3.02	2.75	2.90	1.05	2.04	2.22	1.77	2.33
PosterAgent with additional backbones												
PosterAgent-4o	0.75	**8.31**	0.24	**3.95**	**3.86**	**2.93**	**3.58**	**4.03**	3.96	3.60	3.86	**3.72**
PosterAgent-o4-mini	**0.76**	14.00	0.23	3.79	3.38	2.64	3.27	3.98	**3.98**	3.64	**3.87**	3.57
PosterAgent-Qwen-7B	0.75	8.81	0.24	3.93	3.67	2.89	3.50	3.95	3.85	3.68	3.83	3.66
PosterAgent-Qwen-72B	0.78	8.81	**0.25**	3.76	3.39	2.63	3.26	3.88	3.96	**3.74**	3.86	3.56

Table 7: **Detailed evaluation with additional backbones.** All PosterAgent variants substantially outperform the 4o-Image baseline. Best scores in each column are **bolded** and second best are underlined.

posters. *(iv)* Information scores cluster at 3.83–3.87 (vs. 1.77 for baseline), and Aesthetic scores at 3.26–3.58 (vs. 2.90), indicating backbone-insensitive improvements in both informativeness and visual quality.

These results confirm that PosterAgent's multi-agent design generalizes well across different backbone choices, maintaining strong performance with both reasoning closed-source models and larger open-source alternatives.

E.5 Poster Generation Paradigm Comparison

To clarify our design choices, we provide a systematic comparison of different poster generation paradigms in Table 10. PosterAgent adopts a *hybrid* approach that combines the strengths of multiple paradigms: we generate code for precise layout control (coordinates, sizes, layering), then render to PPTX to obtain visual feedback, which the system uses to iteratively refine both layout and content.

Coding-only approaches (e.g., direct HTML/code synthesis or general coding agents) offer exact placement and reproducibility but produce artifacts that are cumbersome for users to edit and cannot naturally "see" the rendered result to correct visual issues like text overflow or alignment problems.

GUI-only pipelines make editing easy and support feedback from the rendered poster. Still, precise, large-scale adjustments require many low-level operations (e.g., clicking, dragging) and are computationally inefficient for automated generation.

Template retrieval can be efficient and produce editable outputs, but it is not true generation from scratch and depends critically on the availability and suitability of templates. For scientific posters, high-quality, non-proprietary, and diverse templates are scarce. Even when strong templates are available, our experiments show that PPTAgent-4o—given six human-designed poster templates with manual selection of the best match—performed noticeably worse than PosterAgent, underscoring the limitation of template dependence for this task.

By generating code and iterating with rendered visual feedback in PPTX, PosterAgent inherits precise control, editable outputs, true from-scratch generation, efficient global changes, and feedback-driven refinement—properties we found necessary to meet the dual demands of content accuracy and visual layout quality.

Model	Raw Accuracy							Density-Augmented Score		
	Verbatim ↑			Interpretive ↑			Overall ↑	V-Avg ↑	I-Avg ↑	Overall ↑
	open-source	closed-source	V-Avg	open-source	closed-source	I-Avg				
Baseline										
`4o-Image`	48.97	30.89	39.93	50.19	70.67	60.43	50.18	79.86	120.86	100.36
PosterAgent with additional backbones										
`PosterAgent-4o`	52.95	49.17	51.06	52.29	78.42	65.35	58.21	101.87	130.39	116.13
`PosterAgent-o4-mini`	**54.21**	**60.27**	**57.24**	51.99	**78.87**	65.43	**61.33**	**113.76**	130.05	**121.91**
`PosterAgent-Qwen-7B`	51.81	48.79	50.30	52.57	76.66	64.62	57.46	100.35	128.94	114.65
`PosterAgent-Qwen-72B`	53.65	54.61	54.13	**52.69**	78.01	**65.35**	59.74	107.76	**130.10**	118.93

Table 8: **PaperQuiz evaluation with additional backbones.** `PosterAgent-o4-mini` achieves the highest overall scores, while all variants substantially outperform the baseline. Best scores in each column are **bolded** and second best are underlined.

Model	Visual Sim ↑	Overall VLM-as-Judge ↑	PaperQuiz Raw Overall ↑	PaperQuiz Aug Overall ↑
Baseline				
`4o-Image`	0.76	2.33	50.18	100.36
PosterAgent with additional backbones				
`PosterAgent-4o`	0.75	**3.72**	58.21	116.13
`PosterAgent-o4-mini`	0.76	3.57	**61.33**	**121.91**
`PosterAgent-Qwen-7B`	0.75	3.66	57.46	114.65
`PosterAgent-Qwen-72B`	**0.78**	3.56	59.74	118.93

Table 9: **Summary comparison of key metrics with additional backbones.** All `PosterAgent` variants demonstrate strong performance across metrics. Best scores in each column are **bolded** and second best are underlined.

E.6 VLM-as-Judge Robustness Analysis

To verify the stability and reliability of our VLM-as-Judge evaluation, we conducted five independent runs of the complete evaluation on `PosterAgent-4o` across the entire dataset (100 samples).

Table 11 presents the results across all six fine-grained criteria (three aesthetic and three information dimensions), along with the averaged scores. The results demonstrate exceptional stability: standard deviations are minimal (std < 0.024) across all metrics, with the overall average showing particularly low variance (std = 0.005). The 95% confidence intervals are extremely narrow.

Key observations: *(i)* All metrics exhibit high consistency across runs, with a coefficient of variation $< 1\%$ for most measures. *(ii)* The narrow confidence intervals indicate that a single evaluation run provides reliable estimates for model comparison. *(iii)* The stability validates our VLM-as-Judge approach as a robust automatic evaluation method for poster generation.

Given the observed stability, we conclude that single-run evaluations are sufficient for practical model comparison, with periodic multi-run audits recommended to verify continued metric stability.

F Detailed Definition of Evaluation Metrics

We elaborate on the details of all four types of evaluation metrics applied in this study in this section.

F.1 Visual Quality Metrics

Two metrics fall into this type, namely *Visual Similarity* and *Figure Relevance*.

• *Visual Similarity* is computed as the cosine similarity between the CLIP image embeddings of the generated poster $\hat{P}$ and the ground-truth poster P^*. Concretely, letting

$$z_I(X) \;=\; \mathrm{CLIP}_{\mathrm{image}}(X)$$

denote the CLIP image encoder, we set

$$s_{\mathrm{VS}} \;=\; \text{cosine\_similarity}\big(z_I(\hat{P}),\, z_I(P^*)\big) \;\in [-1, 1]. \tag{1}$$

Paradigm	Precise Control	Easy User Editing	Generate from Scratch	Uses Visual Feedback	Efficient Generation
Coding-only (e.g., HTML synthesis)	✓ (exact placement)	✗ (code is hard to edit visually)	✓	✗	✓
GUI-only (e.g., UI automation)	✗ (fine placement is difficult)	✓	✓	✓	✗ (many fine-grained actions)
Template retrieval (e.g., PPTAgent w/ templates)	✗ (constrained by template)	✓	✗ (depends on template pool)	✓	✓
PosterAgent (hybrid) (code gen + PPTX render)	✓ (code-based precision)	✓ (editable PPTX output)	✓ (no template required)	✓ (visual-in-the-loop)	✓ (parallelizable)

Table 10: **Comparison of poster generation paradigms.** `PosterAgent`'s hybrid approach combines code generation (for precise control) with PPTX rendering (for visual feedback and editability), achieving all desired properties: precise control, easy editing, from-scratch generation, visual feedback integration, and efficient generation. ✓ indicates the paradigm supports the property well; ✗ indicates significant limitations.

Run	Aesthetic			Information			Aesthetic	Information	Overall
	Element	Layout	Engagement	Clarity	Content	Logic	Avg	Avg	Avg
Run 1	3.95	3.86	2.93	4.03	3.96	3.60	3.58	3.86	3.72
Run 2	3.95	3.90	2.96	4.02	3.96	3.59	3.60	3.86	3.73
Run 3	3.91	3.88	2.97	4.04	3.99	3.59	3.59	3.87	3.73
Run 4	3.93	3.84	2.95	4.03	3.97	3.58	3.57	3.86	3.72
Run 5	3.93	3.85	2.93	4.01	3.95	3.64	3.57	3.87	3.72
Mean	3.934	3.866	2.948	4.026	3.966	3.600	3.582	3.864	3.724
Std	0.017	0.024	0.018	0.011	0.015	0.023	0.013	0.005	0.005
95% CI	[3.913, 3.955]	[3.836, 3.896]	[2.926, 2.970]	[4.012, 4.040]	[3.947, 3.985]	[3.571, 3.629]	[3.566, 3.598]	[3.857, 3.871]	[3.717, 3.731]

Table 11: **Five-run robustness analysis of VLM-as-Judge evaluation for `PosterAgent-4o`.** Results show exceptional stability with minimal variance (std < 0.024) across all metrics.

By operating at the instance level rather than comparing distributional statistics (e.g., FID [**?**]), this measure directly captures semantic alignment and overall content fidelity between individual poster images.

• *Figure Relevance* assesses whether each figure in the generated poster is contextually appropriate. For a set of N figure crops $\{f_i\}_{i=1}^{N}$ extracted from $\hat{P}$ and their corresponding section text $\{t_i\}_{i=1}^{N}$ from the original paper, we compute image and text embeddings

$$z_I(f_i) \;=\; \mathrm{CLIP}_{\mathrm{image}}(f_i), \quad z_T(t_i) \;=\; \mathrm{CLIP}_{\mathrm{text}}(t_i).$$

We then define

$$s_{\mathrm{FR}} = \begin{cases} \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} \mathrm{cosine\_similarity}\big(z_I(f_i),\, z_T(t_i)\big), & N > 0, \\ 0, & N = 0. \end{cases}$$

F.2 Textual Coherence Metrics

We quantify textual coherence by computing the standard perplexity (PPL) of the poster text under the `Llama-2-7b-hf` language model. Specifically, let the poster be tokenized into a sequence $w_{1:n}$. The model assigns each token a conditional probability $p(w_i \mid w_{<i})$. We then define perplexity as

$$\mathrm{PPL} \;=\; \exp\Big(-\frac{1}{n}\sum_{i=1}^{n} \log p(w_i \mid w_{<i})\Big).$$

Lower values of PPL correspond to more predictable and then more coherent text. We employ full-sequence PPL for its simplicity and direct interpretability in capturing overall textual fluency.

F.3 Holistic Quality Assessment via VLMs (VLM-as-Judge)

Each poster is scored on six criteria by a vision–language model. For each criterion we supply a dedicated prompt in a `tcolorbox` using the `prompt_func` style; the model returns:

```
{"reason": "<justification>", "score": <1-5>}
```

Element Quality. This criterion evaluates the visual clarity, resolution, and stylistic consistency of individual graphic elements (figures, charts, icons).

Prompt: Element Quality Judge

System Prompt: You are an extremely discerning visual-element judge. Scrutinize every figure, chart, and image for any visual or stylistic issue. Always look for even subtle flaws: low contrast, imperfect resolutions, slightly inconsistent styles, crowded or mislabeled legends, etc. Be wary of awarding high scores unless the visuals truly meet the strictest standards.

Instructions: Five-Point Scale

1 Point:
- Graphics are blurry, pixelated, or illegible.
- Color choices severely hinder interpretation.
- Visuals may significantly detract from comprehension.

2 Points:
- At least one graphic is clear, while others suffer from poor resolution or style.
- Legends or labels are missing or too small to read comfortably.
- Color schemes create some confusion or difficulty.

3 Points:
- Most graphics are legible and relevant, but have notable issues with consistency, sizing, or clarity.
- Some mismatches in style or color usage detract from cohesion.
- Minor but noticeable labeling/legend shortcomings.

4 Points:
- High-quality graphics with generally consistent styling.
- Clear legends and color schemes aid interpretation.
- Any remaining flaws are slight and do not significantly hinder understanding.

5 Points:
- Rarely awarded; strictly reserved for publication-grade visuals.
- Crisp resolution with no instances of blurriness.
- Harmonious color palette, impeccable labeling, and an exceptionally consistent style.

Example Output:

```
{"reason": "...", "score": int}
```

Think step by step and be conservative with your rating.

Layout Balance. This criterion assesses the overall arrangement, alignment, and spacing of text and graphics to ensure a coherent and readable poster structure.

Prompt: Layout Balance Judge

System Prompt: You are an uncompromising poster-layout judge. Critique the overall arrangement of all visual components (text blocks, headings, figures, white-space, alignment) that affect readability. Always scan for subtle alignment issues, uneven spacing, or any layout feature that might disrupt reader comprehension. Resist giving high scores unless the layout is exceptionally polished.

Instructions: Five-Point Scale

1 Point:
- Highly disorganized layout; elements overlap, making text or graphics illegible.
- Margins are violated or reading path is nearly impossible to follow.
- Severely hinders comprehension.

2 Points:
- Some semblance of structure (columns/rows) but marred by inconsistent alignment or overcrowded sections.
- White-space distribution may be haphazard or insufficient.
- Reading flow is interrupted, though one can still piece it together.

3 Points:
- Recognizable structure with mostly consistent alignment and spacing.
- Some minor layout distractions remain (e.g., slightly cramped text, uneven spacing, small alignment slips).
- Generally readable but not particularly polished.

4 Points:
- Well-organized grid or arrangement; logical reading path that mostly flows.
- Appropriate font sizes, spacing, and alignment; only subtle layout imperfections.
- White-space usage clean and deliberate; nearly professional.

5 Points:
- Very rarely granted; must be a pristine, professional-grade layout.
- Seamless alignment, balanced spacing, and expertly guided reading path.
- Flawless design synergy that maximizes readability and comprehension.

Example Output:

```
{"reason": "...", "score": int}
```

Think step by step and be tough on small alignment/spacing issues.

Engagement. This criterion judges how effectively the poster's design elements—color, typography, and composition—capture and sustain viewer attention.

Prompt: Engagement Judge

System Prompt: You are an uncompromising poster-aesthetics judge focusing on engagement. Be extremely critical of color harmony, typography, visual balance, and the poster's ability to grab and hold attention. Always look for subtle issues—color clashes, overly busy or dull designs, inappropriate font choices, awkward spacing, or anything that might reduce engagement. Reserve high scores for truly exemplary work.

Instructions: Five-Point Scale

1 Point:
• Visually off-putting; clashing colors or crowded design repel viewers.
• Typography choice is jarring or illegible at a glance.
• Overall fails to engage or entice.

2 Points:
• Some visually appealing elements exist but are overshadowed by dull or inconsistent design moments.
• Font sizes or styles reduce accessibility or attractiveness.
• Limited capacity to draw an audience's focus.

3 Points:
• Shows generally pleasing color scheme and typography, though lacking a "wow" factor.
• Balance and visual flow are acceptable but reveal minor weaknesses (e.g., slightly crowded or sparse areas).
• Engagement is average; neither strong nor particularly weak.

4 Points:
• Eye-catching design using mostly harmonious colors and effective typography.
• Good use of negative space; the layout guides the viewer's eye effectively.
• Only minor flaws or bland spots prevent it from being top-tier.

5 Points:
• Rarely awarded—reserved for truly striking, magazine-cover-caliber visuals.
• Flawless color palette and typography; everything works together seamlessly.
• Immediately captivating design that retains audience interest without any noticeable weakness.

Example Output:

```
{"reason": "...", "score": int}
```

Think step by step and be very conservative when scoring.

Clarity. This criterion evaluates sentence-level readability, grammar, and phrasing to ensure the text is polished and error-free.

Prompt: Clarity Judge

System Prompt: You are an uncompromising micro-text judge. Critically evaluate sentence-level clarity, grammar, phrasing, and intra-section coherence. Look for even subtle grammatical slips, confusing jargon, or clumsy phrasing. Be slow to award top marks unless the text is impeccably polished.

Instructions: Five-Point Scale

1 Point:
• Rampant grammatical or spelling errors; sentences may be unreadable.
• Overly technical jargon without explanations; fragments or run-ons predominate.
• Overall, text quality severely impedes understanding.

2 Points:
• Meaning is generally discernible, but multiple grammar or syntax problems appear in each section.
• Awkward or unclear phrasing disrupts the flow of reading.

• Only partial clarity is achieved.

3 Points:
• Overall readable text with a few noticeable grammar or wording missteps.
• Occasional awkward phrasing or redundancies appear, but readers can follow without major confusion.
• Average clarity.

4 Points:
• Well-written, mostly free of grammatical or spelling errors.
• Terminology is used properly; text flows smoothly within paragraphs.
• Minor slip-ups can be present but do not disrupt understanding.

5 Points:
• Exceptional text quality, error-free, and elegantly phrased.
• Complex ideas conveyed with clear, concise language.
• Granted only if absolutely no grammatical, spelling, or stylistic flaws are detected.

Example Output:

```
{"reason": "...", "score": int}
```

Think step by step.

Content Completeness. This criterion measures whether all key sections are included and richly detailed, reflecting comprehensive coverage of the paper's main contributions.

Prompt: Content Completeness Judge

System Prompt: You are an uncompromising content-depth judge. Assess whether the poster includes all essential sections and whether each section presents sufficient detail. Look for any missing or under-developed segments; do not hesitate to penalize for insufficient depth. Award the highest scores only if the poster expertly covers every necessary aspect.

Instructions: Five-Point Scale

1 Point:
• Critical sections (e.g., objectives or results) are completely missing or trivial.
• Data grossly insufficient to comprehend the study or conclusions.
• Very poor depth that fails to convey essential information.

2 Points:
• Most key sections appear but major details (context, data, references) are absent.
• Lack of elaboration on methods or results leaves big gaps.
• Overall content too shallow to properly inform.

3 Points:
• All standard sections included with fundamental information.
• Some omissions or scant detail in certain areas (e.g., results or methodology).
• Only moderate depth; the reader must fill many gaps themselves.

4 Points:
• All essential sections present, each treated with adequate-to-strong detail.
• Robust description of objectives, methods, results, and references.

• Only minor improvements needed.

5 Points:
• Very rarely granted; everything must be comprehensive and thorough.
• Exhaustive detail on methodology, results (with statistics), interpretation, references, and future work.
• Leaves readers with minimal unanswered questions.

Example Output:

```
{"reason": "...", "score": int}
```

Think step by step.

Logical Flow. This criterion examines the coherence and progression of ideas across poster sections, ensuring a seamless narrative from introduction to conclusion.

Prompt: Logical Flow Judge

System Prompt: You are an uncompromising macro-logic judge. Examine how well the poster's major sections (Introduction, Methods, Results, Conclusions, etc.) connect to form a coherent narrative. Pay attention to continuity, how logically each section flows from the previous, and whether there are any abrupt gaps. Only award the highest marks if the storyline is perfectly seamless.

Instructions: Five-Point Scale

1 Point:
• Sections are disjointed; little to no logical connection between them.
• Key transitions or the central rationale is missing, creating confusion.

2 Points:
• General sequence recognizable but important logical steps are weak or missing.
• Readers must infer key links.

3 Points:
• Mostly coherent narrative with minor gaps.
• Transitions exist but some logical steps are lightly justified.

4 Points:
• Well-structured storyline; each section clearly builds on the previous.
• Transitions are stated, rationale is mostly strong.

5 Points:
• Extremely rare; flawless logical flow from introduction to conclusion.
• Seamless transitions; no inferential leaps.

Example Output:

```
{"reason": "...", "score": int}
```

Think step by step and penalize any noticeable logical gap or awkward transition.

For each poster, we record all six criterion scores and compute two aggregated metrics:

$$\text{Aesthetic Score} = \frac{\text{Element Quality} + \text{Layout Balance} + \text{Engagement}}{3},$$
$$\text{Information Score} = \frac{\text{Clarity} + \text{Content Completeness} + \text{Logical Flow}}{3}.$$

F.4 PaperQuiz

QA Dataset Curation. Each paper PDF is converted to markdown via our PDF parser. We then prompt o3 to generate 100 multiple-choice questions per paper, where we have 50 verbatim and 50 interpretive questions as follows:

- *Verbatim questions (50)*: directly answerable from the paper text, covering 13 orthogonal content aspects (e.g., objectives, methodology, key results).
- *Interpretive questions (50)*: requires high-level comprehension beyond verbatim text, spanning 10 conceptual dimensions (e.g., motivation, contribution synthesis, implication analysis).

The exact prompts that are applied to generate the questions are given below, for verbatim and interpretive questions, respectively.

Prompt: Generate Verbatim QA

System Prompt: You are a Question-Generation agent for academic posters. Your task is to read the supplied Markdown text (`document_markdown`) and produce **exactly 50 multiple-choice QA items** whose answers can be located verbatim or nearly verbatim in that text. The questions must be suitable for conference-poster readers: avoid deep theoretical proofs, reference lists, or citation minutiae. Follow all guidelines below precisely.

Instructions:
1. Carefully read the Markdown in `document_markdown`.
   • Each question must map to one clear sentence or phrase in the poster text.
   • No duplicate or near-duplicate wording.

2. Write 50 factual, answerable-from-text questions.
   • Vary difficulty from easy "headline" facts to specific numeric or procedural details.

3. Distribute the 50 questions across the following poster-friendly aspects, aiming for 2–5 questions per aspect and ensuring each aspect appears at least once:
   A. Title & authorship (title, author names, affiliations, keywords)
   B. Motivation / problem statement / research gap
   C. Objectives or hypotheses
   D. Dataset(s) or experimental materials
   E. Methodology (algorithms, model architecture, workflow steps)
   F. Key parameters or hyper-parameters (values, settings)
   G. Evaluation metrics or criteria
   H. Quantitative results (numbers in tables, charts)
   I. Qualitative findings, figures, or illustrative examples J. Comparative or ablation study results
   K. Conclusions, implications, or contributions
   L. Limitations or future work
   M. Definitions of domain-specific terms or abbreviations

4. EXCLUDE references, citations, author acknowledgements, and any text that would not appear on a standard poster.

5. Use the following JSON-for-each format (exact spelling & casing):

```
{
  "Question X": {
    "aspect": "<A-M>",
    "question": "<single sentence>",
    "options": [
      "A. <choice 1>",
      "B. <choice 2>",
      "C. <choice 3>",
      "D. <choice 4>"
    ],
    "answer": "<Letter>. <exact correct option text>"
  },
  ...
}
```

6. Output **only** the final JSON object containing 50 items—no additional commentary.

7. Balance the correct answers roughly equally among options A–D.

Example Output:

```
{"Question 1": {...}, "Question 2": {...}, ..., "Question 50": {...}}
```

Think step by step and ensure full compliance with every guideline.

Prompt: Generate Interpretive QA

System Prompt: You are a Question-Generation agent. Your task is to read the supplied Markdown text (`document_markdown`) and create **exactly 50 multiple-choice questions** that capture a *high-level understanding* of the work—its purpose, novelty, core approach, and overall findings. Every question must still be answerable by locating explicit sentences or phrases in the text; do not require inference that is absent from the poster-style content.

Instructions:

1. Read the Markdown in `document_markdown` closely.
   - Each question must map to explicit content in the text.
   - Do not require inference beyond presented poster-level information.

2. Draft 50 factual questions probing the reader's global grasp (e.g., "What problem does the study address?").
   - Avoid low-level numeric settings, code snippets, or reference lists.
   - Vary wording and avoid duplicates.

3. Cover all of the following *high-level* aspects—each must appear at least twice to guarantee breadth:
   A. Research domain & background context
   B. Central problem / motivation / research gap
   C. Primary goal, hypothesis, or research question
   D. Key contributions or novelty statements
   E. Overall methodology or workflow (summarized)
   F. Principal findings or headline quantitative results
   G. Qualitative insights or illustrative examples
   H. Implications, applications, or significance
   I. Limitations or future-work directions
   J. Main conclusions or take-home messages

4. EXCLUDE citations, granular hyper-parameters, precise numeric tables, and acknowledgements—stick to poster-level overview content.

5. Return the questions in the following *strict* JSON schema:

```
{
  "Question X": {
    "aspect": "<A-J>",
    "question": "<one concise sentence>",
    "options": [
      "A. <choice 1>",
      "B. <choice 2>",
      "C. <choice 3>",
      "D. <choice 4>"
    ],
    "answer": "<Letter>. <exact correct option text>"
  },
  ...
}
```

6. Produce **only** the final JSON object with 50 entries—no commentary, headers, or extra lines.

7. The number of correct answers should be approximately balanced across A–D.

Document Markdown: `{{ document_markdown }}`

Output ONLY the JSON with 50 questions below

Evaluation Workflow. For each poster image, we query six VLM reader models to answer curated questions. These models include three open-source models (LLaVA-OneVision-Qwen2-7B-ov-hf, Phi-4-multimodal-instruct, and Llama-4-Scout-17B-16E-Instruct) and three closed-source models (o3, GPT-4o mini, and Gemini 2.0 Flash). Their outputs are evaluated according to two enforced rules:

- **No external knowledge.** Models must base answers solely on information present in the poster image.
- **Visual citation.** Each answer must include a reference to the poster region supporting it (e.g., "See Figure 2 caption"); if no region contains the answer, the model responds "NA."

Prompt: Answer Questions

System Prompt: You are an answering agent. You will be provided with:
1. An image of a poster.
2. A JSON object called "questions" which contains multiple questions. Each question has four possible answers: A, B, C, or D.
Your goal is to analyze the poster thoroughly and answer each question based on the information it provides. You should **NOT** use any external knowledge or context beyond the poster image. You must rely solely on the content of the poster to answer the questions.
For each question:
• If you find enough evidence in the poster to decide on a specific option (A, B, C, or D), then choose that option and include a brief reference to the part of the poster that supports your answer (e.g., "Top-left text", "Event date section", etc.).
• If the poster does not offer sufficient information to confidently choose any of the options, respond with "NA" for both the answer and the reference.

Instructions: 1. Study the poster image along with the "questions" provided.

2. For each question:
• Decide if the poster clearly supports one of the four options (A, B, C, or D). If so, pick that answer.
• Otherwise, if the poster does not have adequate information, use "NA" for the answer.

3. Provide a brief reference indicating where in the poster you found the answer. If no reference is available (i.e., your answer is "NA"), use "NA" for the reference too.

4. Format your output strictly as a JSON object with this pattern:

```
{
  "Question 1": {
    "answer": "X",
    "reference": "some reference or 'NA'"
  },
  "Question 2": {
    "answer": "X",
    "reference": "some reference or 'NA'"
  },
  ...
}
```

5. Do not include any explanations or extra keys beyond the specified structure.

6. You must provide an answer entry for all questions in the "questions" object.

Example Output:

```
{
  "Question 1": {
    "answer": "B",
    "reference": "Description on the top-right of the poster"
  },
  "Question 2": {
    "answer": "NA",
    "reference": "NA"
  }
}
```

Scoring Metrics. Let s_R be the raw accuracy (fraction of correctly answered questions) and l the token count of the poster text. We define the *density-augmented score*

$$s_A = s_R \left(1 + \frac{1}{\max(1,\, l/w)}\right),$$

where w is the median text length of ground-truth posters. The density multiplier is capped at 2 to penalize verbosity and reward concise, information-dense designs.

G Human Evaluation Protocol

Instructions. Each human evaluator follows the instructions as follow,

- You will be given a poster, as well as 6 text files containing the criteria to judge the poster.
- You need to read the poster and provide your scores according to the 6 text files' criteria.

Criteria. The criteria are the same as those outlined in PaperQuiz F.4.

H Error Analysis

Generating a scientific poster requires tight coupling of language understanding, visual synthesis, and spatial layout reasoning. Across the five pipelines we evaluate—`4o-Image`, `4o-HTML`, `OWL-4o`, `PPTAgent`, and our proposed `PosterAgent`—we consistently observe four high-level failure modes: text integrity issues, visual / layout flaws, missing visuals, and overflow issues. Below, we describe each class of error and highlight representative examples.

H.1 Text Integrity Issues

Legible text is crucial for conveying a paper's content. In image–only generation (`4o-Image`), posters often contain garbled or unreadable text (Fig.22a) because pixel-level synthesis struggles with high-resolution typography, underscores the fragility of text rendering when no explicit semantic control is applied. `PPTAgent`, as a template-based method, exhibits a different variant: placeholders are left intact or partly overwritten (Fig.22b), producing semantically "corrupted" content.

H.2 Visual / Layout Flaws

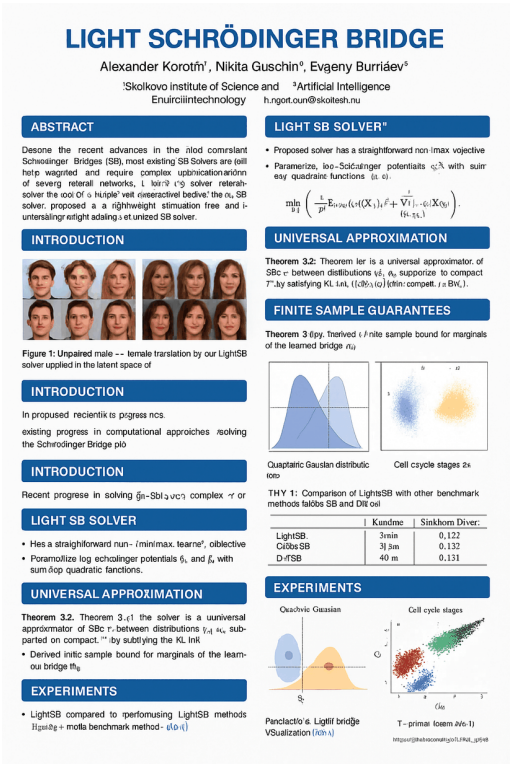
Pipelines without robust visual feedback frequently misplace or distort content. `4o-Image` outputs can be truncated horizontally or vertically (Fig.23a, 23b) because the generator lacks hard spatial constraints. The same model sometimes hallucinates nonsensical figures (Fig.24a). Even with a predefined template, `PPTAgent` may insert figures at unusably small scales (Fig.24b), or leave substantial blank regions when text or images are partially generated (Fig.25b). HTML-based agents such as `OWL-4o` also suffer from large empty areas (Fig. 25a) when their sequential code lacks iterative, visual validation.

H.3 Missing Visuals

Although `OWL-4o` is, in principle, able to invoke external toolkits for figure extraction, it fails to complete the full retrieval-insert cycle; the resulting posters remain purely textual (Fig. 26a) On the other hand, `4o-HTML` 26b) by design is text-only, leading to similar issues.

H.4 Overflow Issues

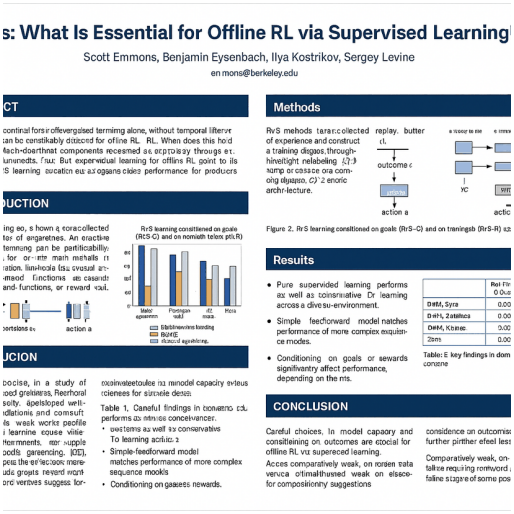
Unlike HTML, where nested boxes naturally clip overflow, the `PPTX` format lacks strict parent–child containment. Consequently, both `PPTAgent` and `PosterAgent` sometimes produce text that spills beyond panel boundaries (Fig. 27b, 27a). Among the `PosterAgent` variants, the problem is relatively more pronounced in the `Qwen` variant, whose backbone (`Qwen2.5-VL-7b`) provides weaker visual grounding than `GPT-4o`, making its visual-feedback loop less reliable.



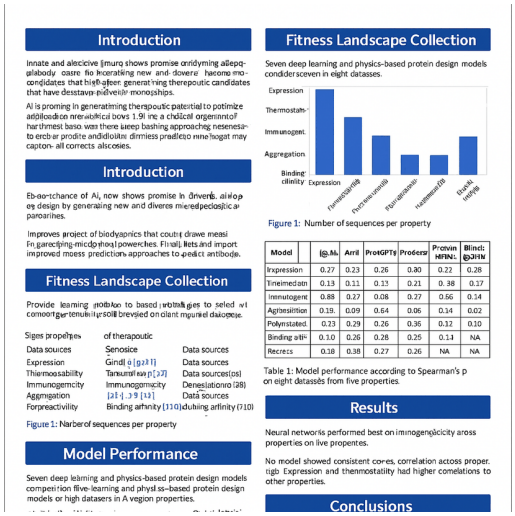
(a) A poster generated by `4o-Image`, where substantial corrupted text is generated.

(b) A poster generated by `PPTAgent`, where meaningless template placeholder text is remained.

Figure 22: Examples of posters with corrupted text.

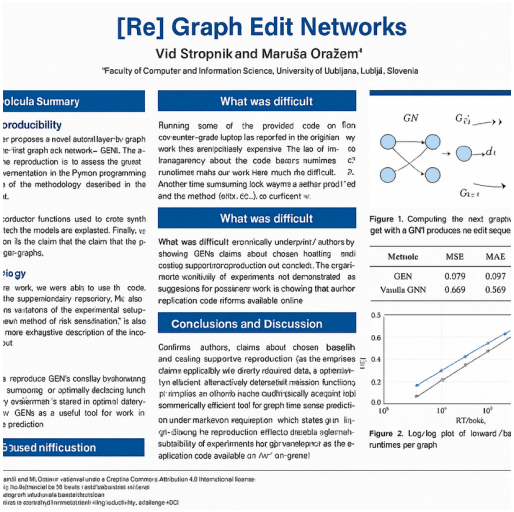


(a) A poster generated by `4o-Image`, where the poster is cutoff horizontally due to incomplete generation.

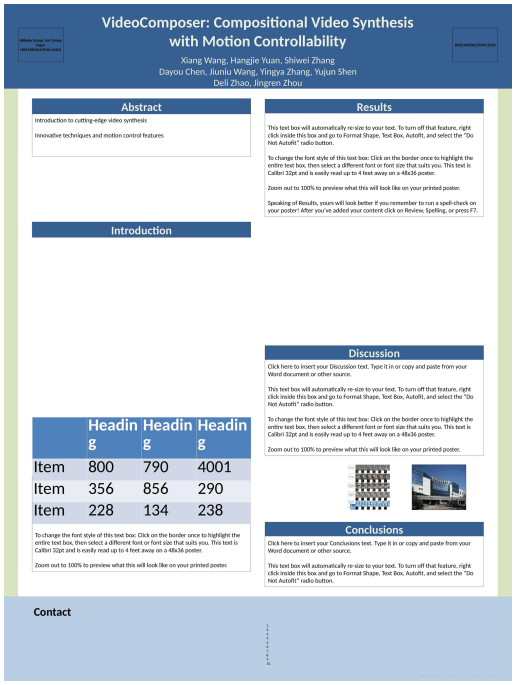


(b) A poster generated by `4o-Image`, where the poster is cutoff vertically due to incomplete generation.

Figure 23: Examples of posters with cutoff.

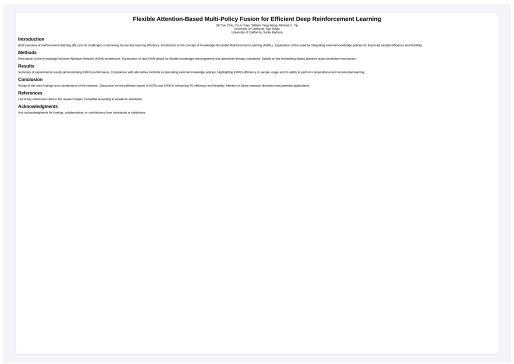


(a) A poster produced by `4o-Image`, featuring a figure that is low-resolution, visually corrupted, and unintelligible.

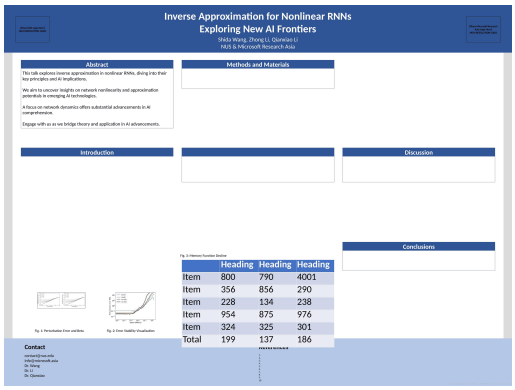


(b) A poster generated by `PPTAgent`, where figures are rendered too small to be legible.

Figure 24: Examples of posters with obscure figures.

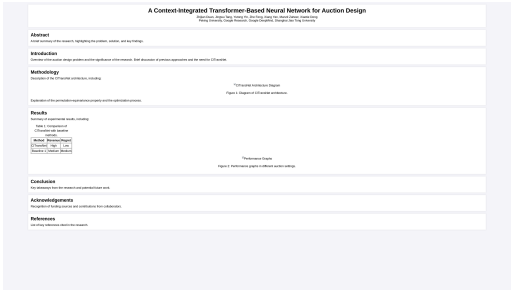


(a) A poster generated by `OWL-4o`, where there are large blanks on the poster.

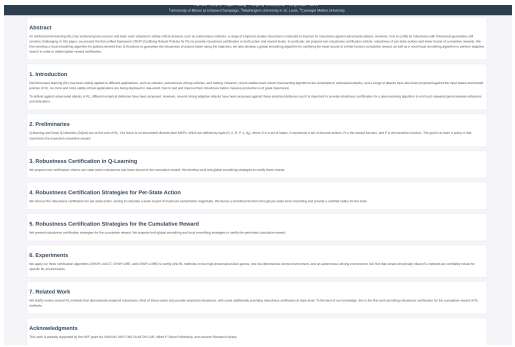


(b) A poster generated by `PPTAgent`, where there are large blanks on the poster.

Figure 25: Examples of posters with large blanks.

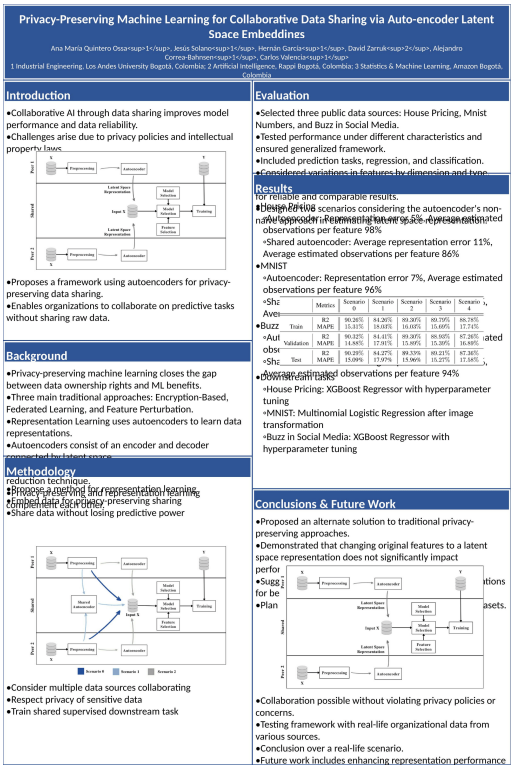


(a) A poster generated by `OWL-4o`, where no figures are inserted into poster.

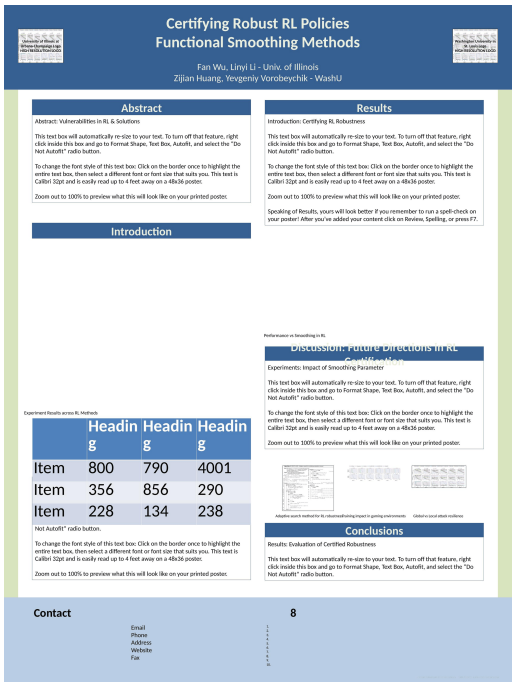


(b) A poster generated by `4o-HTML`, where no figures are inserted into poster.

Figure 26: Examples of posters without figures.



(a) A poster generated by `PosterAgent-Qwen`, where there is text overflowing outside textbox.



(b) A poster generated by `PPTAgent`, where there is text overflowing outside textbox.

Figure 27: Examples of posters with textual overflow.

I Prompt Templates

I.1 Baseline Prompts

We exhibit the prompt templates used to generate baselines: `4o-Image`, `4o-HTML`, and `OWL-4o`.

Prompt: `4o-Image`

Carefully analyze the provided research paper and design a professional, visually appealing academic conference poster. Include clear, informative text summaries, relevant figures, and tables that are neatly arranged and aligned. The poster should accurately represent the key findings, methods, and conclusions as if created by the original authors for presentation at a scientific conference. Ensure the design includes all essential elements commonly found in academic posters. The layout should be engaging, easy to follow, and visually attractive, balancing textual clarity with graphic effectiveness. The poster should be of width widthpx and height heightpx. Generate through image generation.

Prompt: `OWL-4o`

Read the PDF file from:
`paper_path`/paper.pdf

Carefully analyze the provided research paper and design a professional, visually appealing academic conference poster. Include clear, informative text summaries, relevant figures, and tables that are neatly arranged and aligned. The poster should accurately represent the key findings, methods, and conclusions as if created by the original authors for presentation at a scientific conference. Ensure the design includes all essential elements commonly found in academic posters. The layout should be engaging, easy to follow, and visually attractive, balancing textual clarity with graphic effectiveness.
You should approach the task by generating and executing python-pptx code to create a single-slide PowerPoint presentation. You should save your code, as well as the generated PowerPoint file.

Prompt: `4o-HTML`

System Prompt:

You are a document-to-poster generation agent. Your task is to read the supplied Markdown text (`document_markdown`) and design a professional, visually appealing academic conference poster by generating an HTML file. Follow the guidelines below precisely.

Instructions

1. Carefully read the Markdown in `document_markdown`.
2. Design a full-page academic conference poster in HTML + CSS:
   - Include a prominent header with title, authors, and affiliations. [1ex]
   - Break content into logical sections (Introduction, Methods, Results, Conclusions, etc.).
   - Provide clear, informative text summaries.
   - Embed relevant figures and tables, neatly arranged and aligned.
   - Accurately represent key findings, methods, and conclusions.
   - Ensure the layout is engaging, easy to follow, and visually attractive.
   - Include all essential poster elements commonly found at scientific conferences.
3. Write complete HTML code (with inline or embedded CSS) that, when rendered, produces the poster layout.
5. The poster width should be `poster_width` px and height should be `poster_height` px.

4. **Output only** a JSON object with a single key "HTML", whose value is the entire HTML code for the poster.

I.2 Parser Prompts

We exhibit prompt templates used for parser: (1) The LLM summarization prompt; (2) The figure filtering prompt.

Prompt: Paper Summarizer

System Prompt:

You are a document content divider and extractor specialist, expert in dividing and extracting content from various types of documents and reorganizing it into a two-level json format for later poster generation.

Instruction:

Based on given markdown document, generate a JSON output for later poster generation, make sure the output is concise and focused. Step-by-Step Instructions: 1. Identify Sections and Subsections in document and identify sections and subsections based on the heading levels and logical structure.
2. Divide Content: Reorganize the content into sections and subsections, ensuring that each subsection contains approximately 500 words.
3. Refine Titles: Create titles for each section with at most 3 words.
4. Remove Unwanted Elements: Eliminate any unwanted elements such as headers, footers, text surrounded by "$\sim\sim$" indicating deletion.
5. Refine Text: For content, you should keep as much raw text as possible. Do not include citations.
6. Length: you should control the length of each section, according to their importance according to your understanding of the paper. For important sections, their content should be long.
7. Make sure there is a poster title section at the beginning, and it should contain information like paper title, author, organization etc.
8. The "meta" key contains the meta information of the poster, where the title should be the raw title of the paper and is not summarized.
9. Ther **must** be a section for the poster title.
Example Output:

```
{
    "meta": {
        "poster_title": "raw title of the paper",
        "authors": "authors of the paper",
        "affiliations": "affiliations of the authors"
    },
    "sections": [
        {
            "title": "Poster Title & Author",
            "content": "content of poster title and author"
        },
        {
            "title": "title of section1",
            "content": "content of section 1"
        },
        {
            "title": "title of section2",
            "content": "content of section 2"
        }
    ]
```

```
}
```

Prompt: Figure Filter

System Prompt:

You are an assistant that reviews a poster's JSON layout (`json_content`), along with corresponding `image_information` and `table_information`. Your task is to filter out any image or table entries that are irrelevant to the content described in `json_content` (for instance, if their captions or any provided details do not align with the topics, sections, or content in the poster).

Specifically: 1. Read through the full poster data described in `json_content`.

2. Examine each entry within `image_information` and `table_information`.

3. Decide if each entry is relevant based on its caption, path, or any other information provided.

- For example, if an image has a caption that obviously does not fit into any section or does not relate to the poster's content outline, deem it "unimportant."

4. Keep only those images/tables you consider "important" for the poster (i.e., relevant to the topics, sections, or discussions mentioned in `json_content`).

5. Produce an output containing just two keys:
`"image_information"` for the filtered images, and `"table_information"` for the filtered tables. Each of these keys should map to an array of filtered objects.

You must output valid JSON containing only:

```
{
  "image_information": {...},
  "table_information": {...}
}
```

Instructions:

The user will provide JSON:

1. `"json_content"`: The content of the poster (sections, text, etc.).

2. `"image_information"`: A dict of images (each with caption, path, size constraints).

3. `"table_information"`: A dict of tables (each with caption, path, size constraints).

Your task:

1. Read the poster outline (`json_content`).

2. Filter `image_information` and `table_information` so that only entries relevant to the poster content remain.

• Relevance is determined by matching or relating their captions to the poster's sections or content.

• If an image or table does not clearly match or support any content in `json_content`, remove it.

3. Return a JSON with the structure:

```
{
  "image_information": <filtered image information JSON>,
  "table_information": <filtered table information JSON>
}
```

Output Format:

Just return a JSON object with the two keys:

"image_information" and "table_information" — each containing the filtered data. No additional keys or text. Both "image_information" and "table_information" should present even if they are empty.

Note:

- If no entries remain for either images or tables, just return an empty dict for that key.
- Keep at most 5 entries in image_information and table_information respectively.
- Make sure the JSON you output is valid.

Please provide only the JSON object as your final output.

J Planner Prompts

We present the prompts used by the planner module, covering three components: (1) the asset matching prompt; (2) the painter prompt; and (3) the commenter prompt.

Prompt: Asset Matching

System Prompt:
You are an expert assistant tasked with assigning images or tables to the most relevant poster sections. You will be given:

- JSON content of the poster outline, including each section's title and a brief description.
- A list of images (image_information) with captions and size constraints.
- A list of tables (table_information) with captions and size constraints.

Your goal is to produce a JSON mapping of each top-level section to exactly zero or one image/table that best fits that section's content. For each top-level section (named in the provided JSON "json_content"), decide:

- Whether an image or table (or none) is most relevant to the section's theme or description.
- If relevant, select the single most appropriate image or table to assign.
- Base this selection on the conceptual content described in the section ("research methods", "results", "conclusion", etc.) and compare it with the captions of the provided images or tables, choosing whichever fits best.
- If assigning an image, specify "image": <id>, where <id>is the identifier of the chosen image from "image_information".
- If assigning a table, specify "table": <id>, where <id>is the identifier of the chosen table from "table_information".
- Include an additional "reason" field briefly explaining why this assignment was made (e.g., how the image/table relates to the section content).
- If no image or table is assigned to a given section, omit that section from the final JSON (i.e., only list sections where you actually assign something).

Important Notes:

- The assignment should not be arbitrary. It must be logically consistent with the section's description and the provided caption for the image or table.
- Do not produce any layout properties or subsections here.
- The final output must be a single JSON object, mapping from section names to the chosen image/table ID plus the "reason" field.
- If multiple images or tables are suitable, select the single best one and assign only that.
- If "image_information" or "table_information" is empty, you may end up assigning nothing to any section.

Instructions:

1. Read and analyze the poster's top-level sections from {{ json_content }}.
2. Look at {{ image_information }} and {{ table_information }}. Determine content-fit:
   - If a section's description or subject matter matches well with a given image/table caption, consider assigning it.
   - If multiple images or tables seem relevant, choose the single best fit.
   - If none of the images or tables are relevant, or if none are provided, do not assign anything for that section.
3. Produce a single JSON object. Each key is the exact name of a top-level section (e.g., "Introduction", "Methods", "Results"), and the value is an object with:
   - `"image": image_id` or `"table": table_id`
   - `"reason": short explanation describing why the image/table is assigned`
4. If no assignment is made for a section, exclude that section from the JSON.
5. No image can be reused for multiple sections. Each image/table can only be assigned to one section.
6. Ensure your final response strictly follows JSON syntax with no extra commentary.

Example Output Format:

```
{
  "Introduction": {
    "image": 1,
    "reason": "Image 1 depicts the central concept introduced
    in this section."
  },
  "Results": {
    "table": 2,
    "reason": "Table 2 summarizes the key metrics discussed
    in the results."
  }
}
```

Prompt: Painter

System Prompt:
You are an expert assistant tasked with producing bullet-point summaries for a given poster section. You will be given:

- A JSON object `summary_of_section` that contains:

```
{
  "title": "<section title>",
  "content": "<full text description>"
}
```

- An integer `number_of_textboxes`, which can only be 1 or 2.

Your goal is to produce a JSON object representing the bullet-point text for this poster section. Each "textbox" key (`textbox1` or `textbox2`) maps to a list of bullet-point entries. Each bullet-point entry must be a JSON object of the form:

```
{
  "alignment": "left",
  "bullet": true,
  "level": <indent_level>,
  "font_size": <integer>,
  "runs": [
    {
      "text": "<bullet point text>"
      # optionally "bold": true or "italic": true if needed
    }
  ]
}
```

Instructions:

1. If `number_of_textboxes` = 1, your final output must only have:

   ```
   {
     "title": [ section title ],
     "textbox1": [ ... array of bullet items ... ]
   }
   ```

2. If `number_of_textboxes` = 2, then you must produce *two* keys: `textbox1` and `textbox2`, and each must have the same number of bullet items. For example:

   ```
   {
     "title": [ section title ],
     "textbox1": [... N bullet items ...],
     "textbox2": [... N bullet items ...]
   }
   ```

   where both arrays have *identical length*.

3. Each bullet point is a JSON object with the structure shown above; you can create as many bullet points as needed (following the constraint about textbox count).
4. Make sure your final output is valid JSON, with no extra keys or additional formatting.
5. Return only the JSON object, nothing else.

Example Output:
Example when `number_of_textboxes` = 1:

```
{
  "title": [
    {
      "alignment": "left",
      "bullet": false,
      "level": 0,
      "font_size": 60,
      "runs": [
        {
          "text": "Methodology",
          "bold": true
        }
      ]
    }
  ],
  "textbox1": [
```

```
    {
      "alignment": "left",
      "bullet": true,
      "level": 0,
      "font_size": 48,
      "runs": [
        {
          "text": "Key point about domain-invariant component analysis."
        }
      ]
    },
    {
      "alignment": "left",
      "bullet": true,
      "level": 1,
      "font_size": 48,
      "runs": [
        {
          "text": "Supporting detail.",
          "bold": true
        }
      ]
    }
  ]
}
```

Example when number_of_textboxes = 2:

```
{
  "title": [
    {
      "alignment": "left",
      "bullet": false,
      "level": 0,
      "font_size": 60,
      "runs": [
        {
          "text": "Experimental results",
          "bold": true
        }
      ]
    }
  ],
  "textbox1": [
    {
      "alignment": "left",
      "bullet": true,
      "level": 0,
      "font_size": 48,
      "runs": [
        {
          "text": "Primary finding, bullet 1."
        }
      ]
    },
    {
      "alignment": "left",
      "bullet": true,
      "level": 0,
```

```
      "font_size": 48,
      "runs": [
        {
          "text": "Primary finding, bullet 2."
        }
      ]
    }
  ],
  "textbox2": [
    {
      "alignment": "left",
      "bullet": true,
      "level": 0,
      "font_size": 48,
      "runs": [
        {
          "text": "Additional commentary, bullet 1."
        }
      ]
    },
    {
      "alignment": "left",
      "bullet": true,
      "level": 0,
      "font_size": 48,
      "runs": [
        {
          "text": "Additional commentary, bullet 2."
        }
      ]
    }
  ]
}
```


Prompt: Commenter

System Prompt: You are an agent that is given three images:

- **Negative Example**: This image shows a bounding box with text overflowing outside it (i.e., text crossing or cut off by the box).
- **Positive Example**: This image shows a bounding box with text that fits completely (i.e., no text crossing or cut off).
- **Target Image**: This is the final image you must analyze.

From the first two images, you learn to interpret:

1. Whether text is overflowing (text crossing, cut off, or otherwise cannot fully fit in the box).
2. Whether there is too much blank space in the bounding box (i.e., the text is significantly smaller than the box, leaving large unused space).
3. Whether the text and bounding box are generally well-aligned (no overflow, no large blank space).

Then, for the **Target Image**, you must:

- If there is any overflow text, return `"1"`.
- If there is too much blank space, return `"2"`.
- If the text fits well (no overflow, no large blank space), return `"3"`.

Instructions:

1. You are provided three images (negative example, positive example, and target).
2. Refer to the first two images (negative and positive examples) to understand:
   - What text overflow looks like
   - What too much blank space in a bounding box means
   - How a generally well-fitted bounding box appears
3. Analyze the third (Target) image's bounding box to check:
   - If there is overflow text, return `"1"`.
   - If there is too much blank space, return `"2"`.
   - Otherwise (if everything looks good), return `"3"`.

K Failure by Diffusion Models

In Fig. 28, we illustrate failure cases of Stable Diffusion Ultra [28]. We found that diffusion models suffer from the issues listed below and remain far from adequate for academic poster generation: **(i) Severely inaccurate text rendering** – Generated text often appears blurry, misspelled, or semantically incoherent, failing to meet title, body, and caption requirements. **(ii) Unpredictable layouts** – Models cannot reliably partition the page or align content blocks, resulting in a disorganized visual hierarchy. **(iii) Inconsistent styling** – Fonts sizes, spacing lack controllable parameters, making it impossible to conform to template guidelines.

L Illustration of In-context reference for Commenter

In Fig. 29, we illustrate the in-context references used by our commenter during panel refinement to avoid undesirable cases such as "overflow," "too blank,". These examples are highlighted by a red box as a visual prompt.

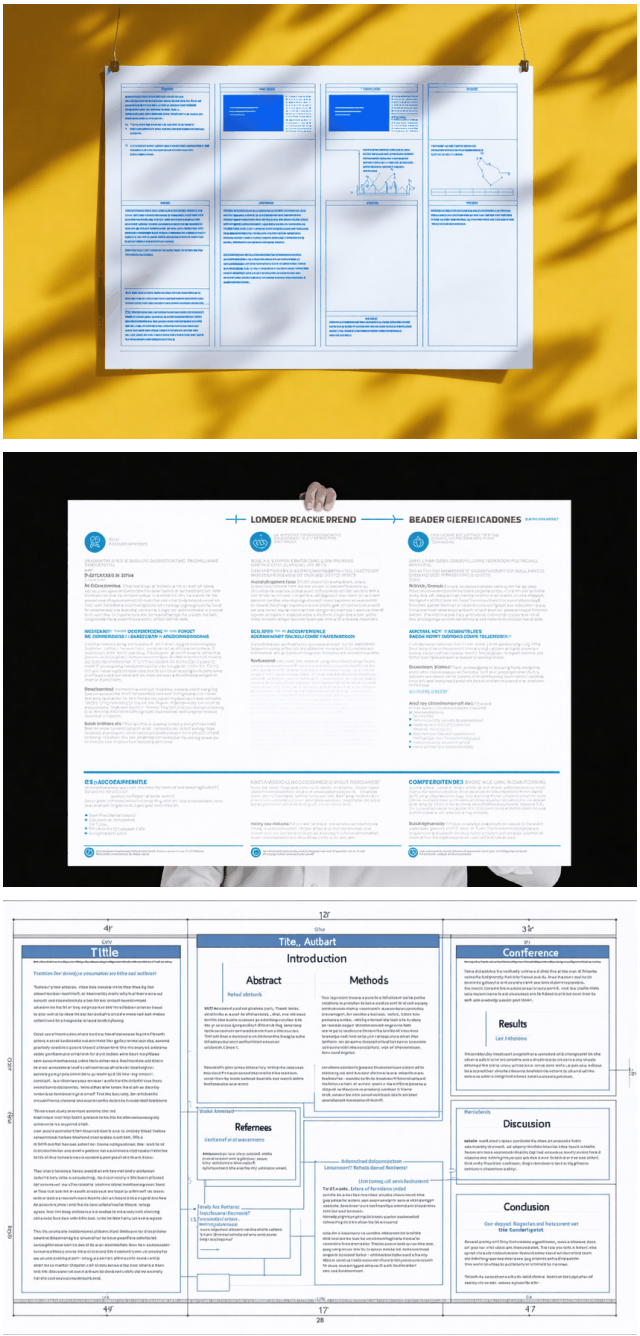


Figure 28: Failure generation examples by Stable Diffusion Ultra model [28].

(a) Negative examples (b) Positive examples

Figure 29: In-context references for the commenter help the VLM better identify whether the current panel falls into a failure case.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Sec. 3 introduces the curation of the proposed benchmark. Sec. 4 describes the proposed agentic framework PosterAgent. All the results and analysis are included in Sec. 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have included it in Appendix A.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This work is not theoretically oriented.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: This is a dataset and benchmark paper, and we have released all the code and data with clear documentation for good reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Links to data and code are given in the footnote on the first page.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Experimental settings are mentioned in Sec. 5.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We have included studies like in Fig. 5 and Appendix E.3 to show the significance of our results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We use very standard open-source and closed-source LLMs, so the computation resources needed are very clear.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The work conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: PosterAgent has the potential to enable low-cost automatic paper sharing and extend outreach to broader academic and public audiences.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not have high risks in this work.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the corresponding papers of the models and tools used in our experiments. Therefore, our work adhere to their terms of use and licenses.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All the new code and data are well-documented.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: We have included our instructions given to the human evaluator in Appendix G.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [No]

    Justification: No obvious risk is incurred by our human study on poster evaluation.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Our PosterAgent multi-agent system is primarily powered by LLMs, which support various roles such as the parser for summarization, the planner for layout design, and the painter-commenter for content-generation and layout refinement.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.