Decoupling Content Delivery from Layout Quality: Diagnosing the Plan–Delivery Gap in Template-Based Slide Generation

Anonymous ACL submission

Abstract

In template-based slide generation, planned content often exceeds the capacity of predefined visual templates, leading systems to omit or compress portions of the source content during rendering. Existing output-only evaluation protocols—whether judge-based, using vision-language models (VLMs), or rule-based geometric metrics—do not directly capture this plan–delivery gap, creating an evaluation blind spot in current benchmarks. To address this, we introduce **PtD-Loss** (Plan-to-Delivery Loss), a reference-free and VLM-judge-free metric that measures the discrepancy between planned visual content and delivered visual elements in rendered slides. As a concrete demonstration of the diagnostic utility of this metric, we additionally propose **SLOT** (**S**ymbolic **L**ayout-c**O**ntent **T**argeting), a framework that manages layout capacity through three rule-grounded modules. On the PSP benchmark, SLOT reduces PtD-Loss from 62.9–64.2% in prior systems to 2.5% while maintaining zero image-pair overlap, and our ablation analysis shows that the modules interact conditionally, not merely additively: structural capacity expansion alone can increase delivery loss, but enables successful redistribution once coupled with downstream spatial feedback. These findings suggest that cumulative ablations and multi-metric evaluation may be useful for analyzing multi-stage agentic generation systems.

1 Introduction

Agentic systems based on large language models (LLMs) have enabled template-based slide generation from source documents. These systems typically follow a plan-then-place pipeline: they first plan the slide content, and then place the planned elements into predefined templates to produce the final presentation (Zheng et al., 2025; Liang et al., 2025). Such pipelines jointly handle document summarization, content structuring, and visual layout, and have shown promising results in recent work. Prior work, however, has centered on the visual quality of the final output, while the fidelity of content delivery in the rendered result remains less examined. In practice, limited template space and layout constraints often force parts of the planned content to be omitted or compressed (Figure 1). Because existing evaluations are largely output-centric, they do not directly capture such plan–delivery gaps. As a result, current systems may exhibit a trade-off between delivery fidelity and layout quality that existing protocols appear insufficiently sensitive to.

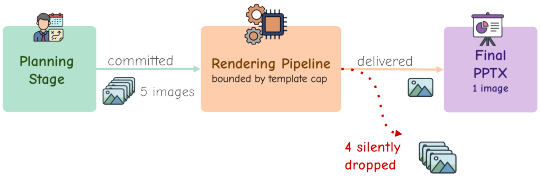

Figure 1: The plan–delivery gap in template-based slide generation: a planning stage commits a set of visual references, but the rendering pipeline, bounded by template capacity, silently drops items that exceed it.

Prior evaluation methods for agentic slide generation fall into two broad categories. The first is rubric-based LLM-as-a-judge evaluation (Zheng et al., 2025; Chen et al., 2026). The second is rule-based evaluation grounded in geometric properties such as overlap and alignment (Liang et al., 2025; Ge et al., 2025). Both remain output-only, and neither directly accounts for discrepancies between planned and delivered content. In analyzing recent systems, we repeatedly observed that visually dense or well-aligned slides received favorable scores despite substantial delivery failures. This observation suggests that current protocols may not adequately reflect delivery fidelity.

To address this limitation, we propose **PtD-Loss (Plan-to-Delivery Loss)**: a reference-free, VLM-judge-free, rule-based metric that measures the discrepancy between planned elements and those ac-

tually delivered in rendered slides. Unlike output-only evaluations, PtD-Loss directly quantifies the mismatch between planning and delivery. The metric is also decomposable into two interpretable components, Planned count and Delivered count, which enables a finer-grained analysis of system behavior that a single aggregated score would otherwise obscure.

We further note that this gap may not be purely an evaluation issue. The plan–delivery gap we observe is consistent with the possibility that certain architectural choices in current systems contribute to delivery failures. In particular, recent systems often treat templates not as rule-grounded structural constraints, but as natural-language context to be interpreted by the LLM. As a result, content delivery and layout stabilization may become entangled within a single generation step, which can lead to omission or excessive compression of the planned content.

To mitigate this issue, we propose **SLOT** (**S**ymbolic **L**ayout-c**O**ntent **T**argeting). SLOT extracts geometric and structural properties from templates as symbolic constraints, and enforces them as explicit constraints throughout the generation pipeline. The key architectural choice is to treat templates as *flexible* rather than *fixed*: when source content exceeds template capacity, SLOT expands the template and reallocates content into the resulting space, rather than dropping source content. By partially decoupling delivery from layout quality, SLOT alleviates the trade-off observed in prior systems. On the PSP benchmark, SLOT reduces PtD-Loss from 62.9–64.2% (baselines) to 2.5%, while maintaining zero image-pair overlap, a combination that no baseline in our comparison achieves.

This paper makes three contributions:

- An analysis of existing evaluation protocols, showing that they may fail to capture plan–delivery gaps in template-based slide generation, together with an empirical observation of a potential trade-off between delivery fidelity and layout quality in current systems.
- **PtD-Loss**, a reference-free and VLM-judge-free rule-based metric that measures discrepancies between planned and delivered content, enabling quantitative analysis of delivery fidelity beyond conventional output-only evaluations.
- **SLOT**, a concrete framework that operationalizes the proposed evaluation perspective as an intervention: by treating templates as symbolic constraints and applying them as explicit, rule-grounded operations throughout the generation process, SLOT serves as a diagnostic case study that exposes the conditional interactions between planning- and execution-stage modules.

This work focuses on template-based slide generation systems that rely on predefined visual structures. Template-free approaches, such as direct code-generation-based slide creation, are outside the scope of this study.

2 Related Work

2.1 Slide Generation

Automatic slide generation has progressed from early text-summarization-based approaches (Sun et al., 2021; Fu et al., 2022; Sefid et al., 2021), through LLM-based multi-stage pipelines (Bandyopadhyay et al., 2024; Cachola et al., 2024; Mondal et al., 2024), toward systems that leverage explicit templates or reference slides. PPTAgent (Zheng et al., 2025) extracts content schemata from reference slides and iteratively edits slides through an editing API. SlideTailor (Zeng et al., 2026) distills implicit preferences from paper–slide pairs and visual templates. AutoPresent (Ge et al., 2025) takes a code-generation approach, directly synthesizing slide code via helper APIs. These template-aware systems commonly treat templates as contextual information interpreted by the LLM. In contrast, our work regards templates as structural constraints that are explicitly enforced.

Several concurrent works pursue related but architecturally distinct directions. SlideGen (Liang et al., 2025) performs outline-based content restructuring with fixed-template generation, and is therefore not directly comparable on benchmarks that condition on a flexible-template interface such as PSP. AeSlides (Pan et al., 2026) trains a slide-generation model with verifiable aesthetic rewards under reinforcement learning; EvoPresent (Liu et al., 2025) similarly relies on a reinforcement-learning aesthetic model (PresAesth) coupled to an iterative self-improvement loop, in the template-free regime. SLOT, in contrast, operates as a training-free, inference-time symbolic framework. DeepPresenter (Zheng et al., 2026) follows a template-free code-generation approach with environment-grounded reflection, which is

outside our template-based scope. We position SLOT as occupying a distinct corner of this design space: *template-aware, training-free, and symbolically constrained.*

2.2 Slide Generation Evaluation

Most existing evaluation methods for slide generation rely on rubric-based VLM-as-a-judge scoring. PPTEval (Zheng et al., 2025) scores Content, Design, and Coherence on a 5-point scale; SlidesBench (Ge et al., 2025) provides both reference-based and reference-free evaluation; and PresentBench (Chen et al., 2026) proposes instance-level, fine-grained checklist-based evaluation. For the related task of PowerPoint task completion, PPTC (Guo et al., 2024) and PPTC-R (Zhang et al., 2024b) introduced rule-based evaluation grounded in API-sequence matching.

While VLM judges can scalably approximate human evaluation, biases such as verbosity, position, and self-preference are widely documented (Zheng et al., 2023; Gu et al., 2024). Recent work also indicates that VLM judges can be insensitive to fine-grained spatial properties such as bounding box collisions (Chen et al., 2024). In the slide generation task specifically, Inui et al. (2025) report that objective visual metrics correlate more strongly with human evaluation than recent VLM judges.

Against this backdrop, rule-based evaluation that does not rely on VLM judges is gaining traction. SlideGen (Liang et al., 2025) introduces Geometry-Aware Density (GAD), a layout-density metric in $[0, 1]$ where higher values indicate more visually balanced use of slide area, computed from the geometric arrangement of rendered shapes. In contrast, our PtD-Loss directly measures the discrepancy between planning and delivery, distinguishing it from existing output-only evaluations.

2.3 Neuro-Symbolic Approaches

Combining rule-based or symbolic constraints with LLM generation is an active research direction, with notable work in layout generation (Zhang et al., 2024a; Chen et al., 2024). In contrast to training-time constraint integration, SLOT proposes a constraint-guided framework that applies template structure at inference time, treats content delivery and layout stability as separate objectives, and applies symbolic constraints across the entire generation pipeline.

The closest concurrent design choice is the outline-refinement mechanism of SlideGen (Liang et al., 2025), in which an LLM agent autonomously splits slides based on semantic length. Our Inspector differs along two architectural axes. First, the splitting signals are visual and structural rather than textual: SPLIT is triggered when a slide is assigned more images than a threshold admits, or when the template is classified as admitting extension, conditions that directly target the image-omission failure mode that PtD-Loss measures. Second, the decisions are symbolically determined rather than left to LLM autonomy, which makes the behavior of the splitting step reproducible across runs.

System	PtD-L. ↓	GAD ↑	Cont. ↑	Des. ↑	Coh. ↑
PPTAgent	62.9%	0.759	3.76	4.05	3.50
SlideTailor	64.2%	0.694	3.52	**4.13**	2.00
SLOT (Full)	**2.5%**	**0.784**	**4.38**	3.66	**4.20**

Table 1: Comparison of PtD-Loss (PtD-L.) with existing output-only metrics. PtD-Loss reveals a larger gap than GAD or PPTEval. Cont./Des./Coh. abbreviate PPTEval's Content, Design, and Coherence scores. GAD on PSP / `gpt-4.1-mini` (N=50); PPTEval (Gemini 2.5 Pro) on a paired N=10 subset.

3 PtD-Loss

3.1 Motivation

Existing evaluation protocols, including rule-based metrics such as GAD (Liang et al., 2025) and VLM-based judges such as PPTEval (Zheng et al., 2025), score what is visible on the final deck without reference to what the system originally intended to place. When a system silently drops items during rendering, the dropped items leave no trace in the scored artifact: the score is computed on an already-filtered output, so two systems with very different delivery behavior can rank similarly—or even in the opposite order—as long as the surviving content lays out cleanly. Table 1 illustrates this blind spot: SLOT differs from the baselines by less than 10% in GAD and 0.6–2.2 points in PPTEval, yet by more than 60% in actual plan–delivery gap. The same limitation applies to human evaluators, since detecting plan–delivery gaps requires direct comparison with the system's planning artifact.

This motivates a complementary measurement that anchors evaluation at the point of plan commitment rather than at the rendered output. To our knowledge, no prior work reports a rule-based, count-level account of how much of the commit-

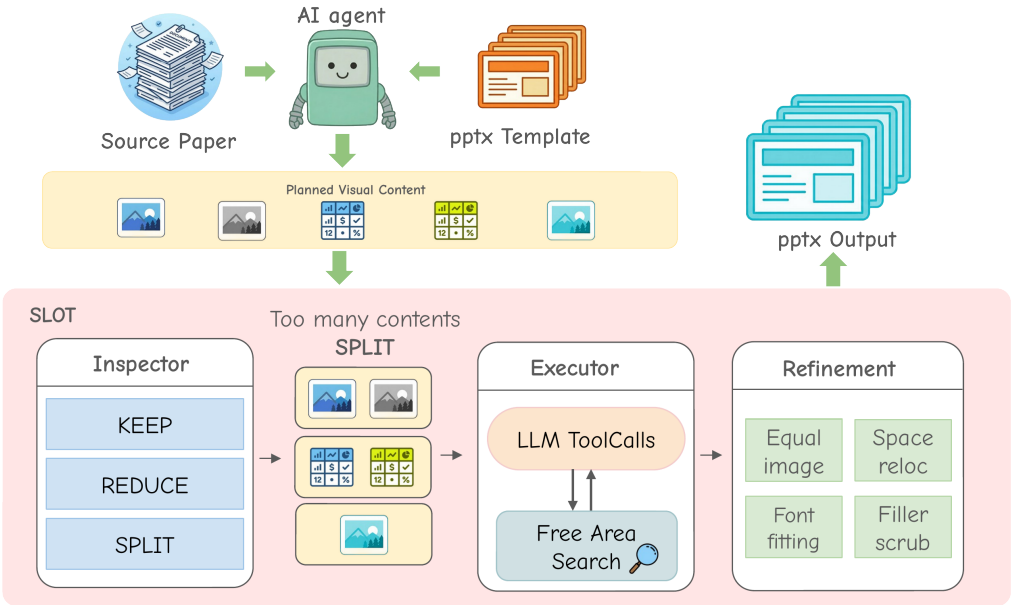

Figure 2: Overview of the SLOT framework. Given a source paper and a pptx template, an LLM agent produces planned visual content; SLOT then processes this plan through three modules: the **Inspector** (pre-LLM, count-level salvage via KEEP/REDUCE/SPLIT; §4.2), the **Executor** (in-LLM Stateful Occupancy Feedback with rule-based free-area search; §4.3), and **Refinement** (post-LLM Pre-render Rule with reclaim slack, image equalization, space reallocation, font fitting, and filler scrub; §4.4).

ted visual plan survives rendering—the gap that PtD-Loss directly quantifies.

3.2 Definition

Agentic slide generation systems typically operate in two stages: planning visual references and rendering them into slides. We use the final planning stage as a common reference point across systems. This stage represents the semantic point of commitment, which allows us to measure how much of the committed plan survives rendering. For a given system i:

- **Planned Count** (P_i): the number of visual references committed at the final planning stage of system i.
- **Delivered Count** (D_i): the number of visual elements successfully instantiated in the rendered output.

PtD-Loss is defined as:

$$\mathcal{L}_{\text{PtD}} = \frac{P - D}{P} \times 100\% \qquad (1)$$

where D denotes the count of visually delivered elements in the rendered slides—picture shapes by default, with native table shapes additionally counted for systems that render tables as native objects (see Appendix A.5). Geometric defects such as overflow, overlap, or truncation constitute an orthogonal failure mode and are evaluated separately using the Overlap-Severity protocol (Appendix B). All measurements are computed by rule from the rendered `.pptx` structure without requiring a VLM judge.

Because $\mathcal{L}_{\text{PtD}}$ is a ratio, a system that plans less can mechanically lower the loss without delivering more; we therefore always report P and D alongside $\mathcal{L}_{\text{PtD}}$ to make this distinction visible (a concrete instance is examined in §5.2).

Conceptually, PtD-Loss can be read as the fraction of committed visual content that the rendering pipeline silently discards. A value near zero indicates that the rendered deck closely realizes the visual plan, regardless of whether that plan was ambitious or modest; a high value indicates that, of everything the system promised to itself during planning, only a fraction reached the final output. This framing is complementary to existing visual-quality metrics: it asks "how much of what the system intended did it actually produce?" rather than "how good does the produced output look?"

Why anchor at the planning artifact. Treating the system's own planning output as the reference is intentional: it sidesteps the under-specification of "ground truth" in slide generation, since multiple presentations of the same paper can be equally valid. PtD-Loss makes no claim about *what* the system should have planned; it only measures whether the system delivers what it itself decided to plan. This places PtD-Loss in a complementary position to semantic metrics, analogous to how test-suite pass rate complements style-based metrics in program synthesis: it does not judge the quality of the plan, only its faithful realization.

3.3 A Possible Structural Source of the Gap

Our analysis of baseline pipelines suggests that this gap may be related to how current systems handle template capacity constraints. When planned content exceeds the available layout capacity, part of the content tends to be omitted during rendering. This behavior appears at two levels. At the per-slide level, both PPTAgent and SlideTailor follow the slot capacity defined by the template layout: for example, 99% of layouts in PSP templates contain at most one PICTURE placeholder (83% contain none and 16% exactly one; derived from layout inspection, Appendix A.1), so an outline that assigns multiple images to such a slide forces the editor to instantiate only one and omit the remaining references. SlideTailor further introduces deck-level constraints through a fixed slide count and a no-reuse policy for images, which together limit the total number of deliverable visual elements. Notably, relaxing the deck-level cap alone would not close this gap, since the per-slide layout capacity would remain binding: a baseline forced to fill more slides would only redistribute the same content more sparsely. These mechanisms appear consistent with the slightly higher PtD-Loss of SlideTailor relative to PPTAgent, while both systems exhibit a similar tendency to prioritize template capacity once layouts become saturated. In Section 4, we describe SLOT, which explores dynamic capacity expansion as one way to mitigate this behavior.

4 The SLOT Framework

4.1 Overview

We introduce SLOT, a slide-generation framework designed to mitigate the structural source of plan–delivery loss identified in §3.3. SLOT also serves as a concrete testbed for evaluating the diagnostic utility of PtD-Loss. Its central design choice is to manage template capacity through explicit, rule-grounded operations rather than relying on the layout schema alone: SLOT extracts symbolic structures (slot coordinates, bounding capacities) from each template, then plans, allocates, and refines content placement under those constraints. When source content exceeds capacity, SLOT expands the available structural boundaries instead of dropping items. Table 2 contrasts this handling with that of the baselines, and Figure 2 provides an overview of the pipeline.

Stage	Baselines	SLOT
Template policy	Fixed	Flexible
Per-slide cap	Schema-bound	Rule-extended
Deck-wide cap†	Fixed + no-reuse	None
Overflow	Truncated	Expanded, reallocated

Table 2: Stage-wise design choices. Baselines (PPTAgent, SlideTailor) rely on fixed-constraint dropping, whereas SLOT couples rule-based template extension with spatial reallocation. †SlideTailor only; PPTAgent has no deck-wide cap.

SLOT addresses this problem through three sequential modules: a pre-LLM Inspector for schema validation and bounded salvage, an in-LLM Stateful Occupancy Feedback loop for decision-time spatial advisories, and a post-LLM Pre-render Rule for rule-based geometric refinement. Their interactions are analyzed in §6 under the term *compensatory absorption*.

4.2 Inspector

Operating between outline formulation and slide-level tool execution, the Inspector constrains the LLM's action space in three ways. First, it bounds tool arguments via schema-level type and range checks. Second, it extracts geometric properties from templates, such as placeholder coordinates, fonts, and margins, and computes slide-level capacity bounds by rule for images, bullets, and characters. Third, when a slide is predicted to exceed these bounds, the Inspector applies one of three bounded salvage actions: **KEEP**, **REDUCE** through paraphrastic compression, or **SPLIT** across child slides. Importantly, SPLIT operates only at the level of *counts*: it decides how many images are assigned to each child slide, but does not commit to where on a slide an image will be placed. The actual spatial placement is deferred to the downstream Stateful Occupancy Feedback module (§4.3). This separation isolates two respon-

sibilities, structural capacity expansion and intra-slide spatial reasoning, into distinct, independently analyzable stages. The full branching rule between KEEP, REDUCE, and SPLIT, together with the per-template SPLIT trigger thresholds and the template classification scheme, is provided in Appendix A.1. Unlike SlideGen's outline-time topic split, which is triggered by semantic length, the Inspector operates on geometric capacity constraints, and its salvage actions later interact with downstream modules in the *compensatory absorption* behavior analyzed in §6.

4.3 Stateful Occupancy Feedback

To bridge planning and execution, we introduce a closed-loop feedback mechanism, the **Executor**. As the LLM emits slide-level tool calls sequentially, the executor maintains the slide's geometric state and performs a rule-based free-area search to identify candidate placement regions. At each step it injects the updated occupancy state, including the set of unoccupied regions and heuristic placement hints, into the LLM's prompt. The mechanism is intentionally soft: occupancy-violating tool calls are not blocked. The occupancy computation, the free-area search procedure, and the associated thresholds are detailed in Appendix A.2.

This division of labor isolates two responsibilities: *where* on a slide each image is placed is decided by the rule-based free-area search, while the LLM retains control over higher-level choices such as which slot to assign next given the current free regions. The mechanism also acts as a recovery channel for the structural expansions produced by the Inspector: empty child slides generated by SPLIT enter the same free-area pipeline, so previously unplaced images can be allocated into their available regions rather than being silently dropped.

4.4 Pre-render Rule

After the LLM finalizes its spatial assignments, the Pre-render Rule, hereafter **Refinement**, applies geometric refinements by rule in place, without adding or removing slides. It performs five operations in sequence: *reclaim slack*, which shrinks oversized text boxes to fit their actual content; *image equalization*, which standardizes dimensions and grid-aligns pictures to remove residual overlaps; *space reallocation*, which shrinks pictures when additional text space is required; *binary-search font fitting*, which finds the largest font size that avoids overflow; and *filler scrub*, which removes unused placeholders and template filler text (e.g., "Click to add title"). Separating stateful feedback from rule-based post-processing preserves the LLM's flexibility while maintaining spatial consistency. A summary of the operations is provided in Appendix A.3; the PSP paper–template pair identifiers are listed in Appendix C.

5 Experiments

5.1 Experimental Setup

Dataset, Baselines, and Backbones. We evaluate on the PSP dataset, comprising 50 paper–template pairs curated from the SlideTailor benchmark (Zeng et al., 2026). We adopt PSP because, to our knowledge, it is the only publicly available and well-curated paper–template pair dataset for template-based slide generation; ad-hoc dataset collection would risk introducing curation bias unfavorable to the baselines. Although $N{=}50$ at the deck level, the effective evaluation surface comprises over 500 rendered slides, providing per-slide measurement granularity. To benchmark performance, we use the official implementations of two recent template-aware systems, PPTAgent (Zheng et al., 2025) and SlideTailor (Zeng et al., 2026), with code modifications limited to backbone replacement. Our primary experiments employ `gpt-4.1-mini` as a shared backbone to isolate architectural differences and ensure cost efficiency. To check that our findings are not backbone-dependent, we conduct supplementary evaluations using `gpt-4o-2024-08-06` (hereafter `gpt-4o` for brevity), the backbone reported in the official releases of both baselines.

Metrics. Our primary metric is PtD-Loss, defined in §3.2. We report Planned count P, Delivered count D, and Loss $\mathcal{L}$. To enable a fair comparison across architectures, we anchor the measurement at the final semantic planning stage unique to each system: post-Inspector for SLOT, the outline phase for PPTAgent, and the layout-matching phase for SlideTailor (detailed anchor mappings in Appendix A.5). As an auxiliary metric for spatial reasoning, we measure the count of image-pair occupancy violations (protocol in Appendix B).

5.2 Main Results

Table 3 reports PtD-Loss for the three systems on the 50 paper–template pairs of PSP, under both backbones (`gpt-4.1-mini` and `gpt-4o`).Under the

System	Backbone	P	D	PtD-Loss ↓	GAD ↑
PPTAgent	`gpt-4.1-mini`	607	225	62.9%	0.759
PPTAgent	`gpt-4o`	356	291[†]	18.3%[*]	0.766
SlideTailor	`gpt-4.1-mini`	371	133	64.2%	0.694
SlideTailor	`gpt-4o`	336	133	60.4%	0.714
SLOT (ours)	**`gpt-4.1-mini`**	**1021**	**995**	**2.5%**	**0.784**
SLOT (ours)	`gpt-4o`	1032	974	5.6%	0.768

Table 3: PtD-Loss and GAD on PSP with N=50 under two backbones. P=Planned, D=Delivered. [†]For PPTAgent under `gpt-4o`, D=291 comprises 247 picture shapes and 44 native table shapes, counted per the convention in Appendix A.5; [*]the apparent drop reflects a 41% decrease in P, not improved delivery (see §5.2).

shared `gpt-4.1-mini` backbone, both baseline systems exhibit substantial plan–delivery discrepancies, yielding PtD-Losses of 62.9% and 64.2%. SLOT reduces this gap to 2.5%. The improvement persists across backbones: under `gpt-4o`, SLOT achieves 5.6% PtD-Loss compared with SlideTailor's 60.4%.

PPTAgent's `gpt-4o` result, 18.3%, presents a notable case worth examining. While its PtD-Loss appears to drop, this reduction is not driven by improved delivery: the LLM generates a substantially shorter plan, P drops by 41%, from 607 to 356, while D increases only modestly, from 225 to 291. The layout ceiling (§3.3) bounds delivery regardless of planner strength: SlideTailor's D stays at 133 across both backbones, and a stronger planner mainly compresses P rather than enlarging D. The P/D decomposition is intended to surface exactly this separation.

5.3 SLOT Ablation

To analyze how each module within SLOT contributes to delivery fidelity and layout quality, we perform a cumulative ablation study. We compare four configurations: (1) **Safety-net only**, in which all SLOT modules are disabled except schema validation; (2) the addition of the Inspector; (3) the subsequent integration of Stateful Occupancy Feedback; and (4) the **Full SLOT** pipeline. Figure 3 reports PtD-Loss with its P/D decomposition, while Figure 4 provides qualitative comparisons. The Safety-net only configuration removes template-level dropping stages and achieves a low PtD-Loss of 1.3%, but without spatial reasoning it produces 1,011 image-pair occupancy violations.

Adding the Inspector substantially changes system behavior. By distributing overflowing image counts across newly generated child slides, the planned count increases from 1,021 to 1,032. However, because the Inspector does not allocate concrete placement regions, many of these additional

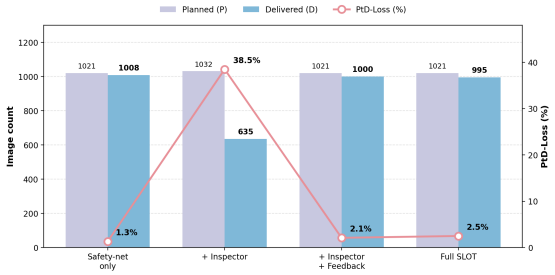

Figure 3: Cumulative ablation of SLOT modules on PSP with gpt-4.1-mini (N=50). Bars show Planned (P) and Delivered (D) image counts; the overlaid line shows PtD-Loss (%).

slots remain unused, reducing D from 1,008 to 635 and increasing PtD-Loss to 38.5%. In this setting, the Inspector alone exhibits a failure pattern reminiscent of truncation-based baselines.

Adding Stateful Occupancy Feedback resolves much of this mismatch by exposing unoccupied regions within the child slides and converting the previously unused slots into concrete placements. As a result, D recovers to 1,000, PtD-Loss decreases to 2.1%, and occupancy violations drop from 1,011 to 110.

Finally, Full SLOT introduces rule-based post-processing through the Pre-render Rule. Although PtD-Loss increases slightly from 2.1% to 2.5%, the remaining occupancy violations are eliminated, producing collision-free layouts.

6 Analysis

6.1 Compensatory Absorption

The P/D decomposition in Figure 3 suggests that the SLOT modules interact conditionally, not merely additively. In particular, the effect of the Inspector appears to depend on whether a downstream module is available to translate its count-level decisions into actual placements. We refer to this conditional recovery behavior as *compensatory absorption*.

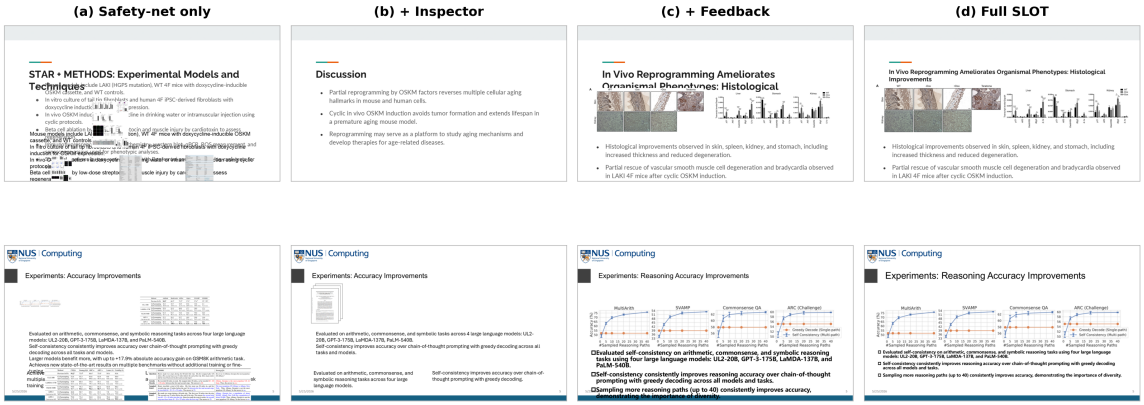

Figure 4: Qualitative comparison of SLOT modes across two papers from PSP, each row showing the same slide rendered under four configurations. The same failure pattern appears across both: (a) Safety-net only produces text–text and text–image overlaps; (b) Inspector alone leaves some child slides underutilized; (c) adding Feedback redistributes content into available regions; (d) Full SLOT eliminates remaining defects through Pre-render Rule refinement.

When the Inspector is applied alone, the planned count increases from 1,021 to 1,032 by distributing overflowing images across newly generated child slides. However, SPLIT only determines how many images each child slide should contain, not where those images should be placed. Without a downstream mechanism for allocating concrete on-slide regions, many of the additional slots remain unused, reducing D from 1,008 to 635 and increasing PtD-Loss to 38.5%. Once Stateful Occupancy Feedback is introduced, its free-area search identifies available regions within the child slides and enables these previously unused slots to be instantiated as actual placements. As a result, D recovers to 1,000 and PtD-Loss decreases to 2.1%.

This interaction also explains why "zero overlap" alone may be insufficient: both Inspector-only and Full SLOT achieve zero occupancy violations in Figure 3, but Inspector-only does so partly because many planned images are never rendered. The P/D decomposition reveals that one of the two well-formed configurations is well-formed in part by virtue of having shed content along the way.

More broadly, this pattern suggests that cumulative ablations and decomposed metrics may be useful whenever planning-stage and execution-stage modules in agentic systems own disjoint sub-objectives, as isolated ablations alone can misrepresent their joint behavior.

6.2 Reinterpreting SLOT's Contribution

A natural interpretation is that SLOT's higher Delivered count D stems from its ability to dynamically increase the number of slides. We view this as one component of a broader architectural shift: PPTAgent and SlideTailor treat templates as *fixed* and truncate on overflow, whereas SLOT *adapts* the template through three rule-grounded operations—structural expansion by the Inspector, spatial allocation by Stateful Occupancy Feedback, and geometric refinement by the Pre-render Rule. The compensatory-absorption analysis (§6.1) indicates that none of these modules alone is sufficient: structural expansion only increases delivery when downstream allocation places the new slots, and stable layout quality requires post-hoc geometric refinement on top of those placements. The contribution is therefore less a single mechanism than a coordinated separation of structural, spatial, and geometric responsibilities, made possible by the rule-grounded interface between modules.

7 Conclusion

We presented **PtD-Loss**, a reference-free and VLM-judge-free metric for plan–delivery gaps in template-based slide generation, together with **SLOT**, a framework that manages delivery fidelity and layout quality through rule-based capacity adaptation. SLOT consistently reduces PtD-Loss on PSP while maintaining low occupancy violations under multiple backbones, and our analysis shows that its modules interact conditionally (*compensatory absorption*)—the Inspector alone increases delivery loss, but behaves differently once coupled with downstream redistribution.

Limitations

We note several limitations. PtD-Loss focuses specifically on delivery fidelity and is intended to complement, rather than replace, semantic or rubric-based evaluation protocols. Our experiments are limited to the 50 paper–template pairs from PSP and to template-aware slide-generation systems; broader validation across additional domains, templates, and model backbones remains future work. Finally, while our cumulative ablation suggests conditional interactions between modules, more exhaustive factorial studies could further characterize these dependencies.

References

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

Isabel Cachola, Silviu Cucerzan, Allen Herring, Vuksan Mijovic, Erik Oveson, and Sujay Kumar Jauhar. 2024. Knowledge-centric templatic views of documents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15460–15476, Miami, Florida, USA. Association for Computational Linguistics.

Jian Chen, Ruiyi Zhang, Yufan Zhou, Rajiv Jain, Zhiqiang Xu, Ryan Rossi, and Changyou Chen. 2024. Towards aligned layout generation via diffusion model with aesthetic constraints. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Xin-Sheng Chen, Jiayu Zhu, Pei-lin Li, Hanzheng Wang, Shuojin Yang, and Meng-Hao Guo. 2026. PresentBench: A fine-grained rubric-based benchmark for slide generation. *Preprint*, arXiv:2603.07244.

Tsu-Jui Fu, William Yang Wang, Daniel McDuff, and Yale Song. 2022. DOC2PPT: Automatic presentation slides generation from scientific documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 634–642.

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. 2025. AutoPresent: Designing structured visuals from scratch. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2902–2911.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hongkun Tan, Xinyu Zhai, Chao Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Yuanzhuo Wang, and Jian Guo. 2024. A survey on LLM-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Yiduo Guo, Zekai Zhang, Yaobo Liang, Dongyan Zhao, and Nan Duan. 2024. PPTC benchmark: Evaluating large language models for PowerPoint task completion. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8682–8701, Bangkok, Thailand. Association for Computational Linguistics.

Tai Inui, Steven Oh, and Magdeline Kuan. 2025. Seeing like a designer without one: A study on unsupervised slide quality assessment via designer cue augmentation. *arXiv preprint arXiv:2508.19289*.

Xin Liang, Xiang Zhang, Yiwei Xu, Siqi Sun, and Chenyu You. 2025. SlideGen: Collaborative multimodal agents for scientific slide generation. *Preprint*, arXiv:2512.04529.

Chengzhi Liu, Yuzhe Yang, Kaiwen Zhou, Zhen Zhang, Yue Fan, Yannan Xie, Peng Qi, and Xin Eric Wang. 2025. Presenting a paper is an art: Self-improvement aesthetic agents for academic presentations. *Preprint*, arXiv:2510.05571.

Ishani Mondal, Shwetha S, Anandhavelu Natarajan, Aparna Garimella, Sambaran Bandyopadhyay, and Jordan Boyd-Graber. 2024. Presentations by the humans and for the humans: Harnessing LLMs for generating persona-aware slides from documents. In *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. Association for Computational Linguistics.

Yiming Pan, Chengwei Hu, Xuancheng Huang, Can Huang, Mingming Zhao, Yuean Bi, Xiaohan Zhang, Aohan Zeng, and Linmei Hu. 2026. AeSlides: Incentivizing aesthetic layout in LLM-based slide generation via verifiable rewards. *arXiv preprint arXiv:2604.22840*.

Athar Sefid, Prasenjit Mitra, Jian Wu, and C Lee Giles. 2021. Extractive research slide generation using windowed labeling ranking. In *Proceedings of the Second Workshop on Scholarly Document Processing*, pages 91–96, Online. Association for Computational Linguistics.

Edward Sun, Yufang Hou, Dakuo Wang, Yunfeng Zhang, and Nancy X. R. Wang. 2021. D2S: Document-to-slide generation via query-based text summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1405–1418, Online. Association for Computational Linguistics.

Wenzheng Zeng, Mingyu Ouyang, Langyuan Cui, and Hwee Tou Ng. 2026. SlideTailor: Personalized presentation slide generation for scientific papers. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence.

Jiahao Zhang, Ryota Yoshihashi, Shunsuke Kitada, Atsuki Osanai, and Yuta Nakashima. 2024a. VASCAR: Content-aware layout generation via visual-aware self-correction. *arXiv preprint arXiv:2412.04237*.

Zekai Zhang, Yiduo Guo, Yaobo Liang, Dongyan Zhao, and Nan Duan. 2024b. PPTC-R benchmark: Towards evaluating the robustness of large language models for PowerPoint task completion. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12387–12402, Miami, Florida, USA. Association for Computational Linguistics.

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

Hao Zheng, Guozhao Mo, Xinru Yan, Qianhao Yuan, Wenkai Zhang, Xuanang Chen, Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. 2026. DeepPresenter: Environment-grounded reflection for agentic presentation generation. *arXiv preprint arXiv:2602.22839*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36.

This appendix summarizes the algorithmic structure, evaluation protocols, and dataset statistics used in our experiments.

A SLOT Implementation Details

A.1 Inspector Branching Algorithm

The Inspector is a two-stage *Plan + Execute* module. The Plan stage examines each slide's content (image count, bullet count, text character count) and chooses one of three actions; the Execute stage rewrites the slide(s) accordingly.

Capacity bounds. The Inspector uses the fixed capacity bounds listed in Table 4. These values are held constant across all experiments.

Threshold	Value	Meaning
MAX_IMAGES_PER_SLIDE	3	image+table count above which SPLIT fires
MIN_IMAGES_FOR_TEXT_OVERFLOW	2	min image count to activate bullet cap
MAX_BULLETS_WITH_IMAGES	4	bullet cap when slide has ≥ 2 images
MAX_TEXT_CHARS_PER_SLIDE	500	body-text character cap
REDUCE_MAX_DROP_FRAC	0.5	if REDUCE drops $> 50\%$ of chars, fall back to SPLIT

Table 4: Fixed capacity bounds used by the Inspector.

The thresholds were selected through small-scale pilot inspection on a held-out subset of paper–template pairs (disjoint from PSP) and kept fixed across all experiments reported in this paper; no per-paper or per-template tuning was performed.

Branching rules. Algorithm 1 formalizes the Inspector's decision matrix. The effective image count, $n_{\text{img}} = |\mathcal{I}| + |\mathcal{T}|$, sums planned images and tables: tables are counted as images because the renderer instantiates both as PICTURE shapes.

Algorithm 1 Inspector branching rule

Require: slide s with image set $\mathcal{I}$, table set $\mathcal{T}$, bullet count n_{bul}, char count n_{char}; thresholds θ_{img}=3, θ_{bul}=4, θ_{char}=500, θ_{min}=2
Ensure: action $a \in \{\text{KEEP}, \text{REDUCE}, \text{SPLIT}\}$

```
n_img ← |I| + |T|                          ▷ effective image count
if s is Cover / TOC / SectionHeader then
    return KEEP                            ▷ structural slide
end if
if n_img > θ_img then
    return SPLIT                           ▷ no atomic exception
else if n_img ≤ θ_min and n_bul > θ_bul then
    return REDUCE                          ▷ drop middle bullets
else if n_char > θ_char and n_img ≤ θ_img then
    return REDUCE                          ▷ paraphrase, no split
else
    return KEEP
end if
```

Effective vs. raw image count. Because tables are PICTURE shapes after rendering, treating them as images is essential for the per-slide cap to predict layout overflow. The Inspector classifies templates implicitly by this density check rather than via a separate "single-image schema" tag; the PSP placeholder-density figures cited in §3.3 (83% zero-PICTURE, 16% single-PICTURE, 1% multi-PICTURE layouts) are derived by inspecting the slot composition of each layout in each template's slide master. The full per-template breakdown is provided in Appendix C.1.

Per-pair action distribution. Across the 50 PSP papers under `gpt-4.1-mini`: KEEP triggers 531 times (median 9/pair, triggered in 50/50 papers), REDUCE 164 times (median 3/pair, 46/50 papers), SPLIT 84 times (median 1/pair, 36/50 papers). SPLIT is therefore a regular operation rather than an edge case; approximately 72% of papers exercise it at least once.

A.2 Stateful Occupancy Feedback

Stateful Occupancy Feedback (SOF) maintains slide-level occupancy state across tool calls and injects updated state into the LLM conversation as user messages.

Bounding box computation.

```
norm_rect = [shape.left / slide_width,
             shape.top  / slide_height,
             shape.width  / slide_width,
             shape.height / slide_height]
```

All coordinates are normalized to slide dimensions (0–1).

Free-area search. Maximal-rectangle algorithm via histogram + monotone stack:

1. Build a 40×40 occupancy grid; each cell is approximately 2.5% of slide width and height.
2. Mark occupied cells, with a 2% buffer around each occupied shape.
3. For each row, build a histogram of consecutive free cells above; find the largest rectangle in histogram via monotone stack.
4. Return up to 3 free areas above thresholds: width $\geq 15\%$ of slide, height $\geq 10\%$ of slide.

Advisory injection format.

```
**Updated Occupancy after step N:**
{
  "occupied": [
    {"shape_id": int, "rect": [x, y, w, h],
     "description": str}
  ],
  "suggested_free_areas": [
    {"x": 0..24, "y": 0..24,
     "w": 0..24, "h": 0..24,
     "description": str}
  ],
  "hint": "Use suggested_free_areas to avoid overlap"
}
```

Coordinates inside the advisory use a 25-cell scale (0–24). The previous snapshot is overwritten each tool call to bound context growth.

Soft enforcement. The advisory is informational only: LLM tool calls that violate `suggested_free_areas` are not blocked. Residual violations are resolved by the Pre-render Rule (§A.3).

A.3 Pre-render Rule

The Pre-render Rule applies lightweight geometry-aware corrections after tool execution, comprising five conceptual operations in sequence:

1. **Reclaim slack**: tighten text boxes whose allocated height exceeds the content they actually hold.
2. **Image equalization**: when multiple pictures share a slide, normalize their sizes and grid-align them along the largest available region.
3. **Proportional space redistribution**: when text would not fit at a minimum legible size, shrink pictures proportionally to free vertical space for text.
4. **Font fitting**: binary-search the largest font size that keeps each text frame within its bounds.
5. **Residual overlap correction**: scrub unused placeholders and template filler text, and flag any remaining hard overlaps under a strict IoU threshold.

We provide the algorithmic structure and configuration used in our experiments; lower-level implementation details are omitted for brevity.

A.4 Prompts

We provide abbreviated descriptions of the prompt templates used by the Inspector module. The excerpts below capture the task framing and constraints used in the main prompt stages; full verbatim prompts are omitted for space.

Inspector Plan Prompt. The Plan stage receives a JSON array of overcrowded slides (each with title, effective image count, bullet count, character count, and content) and returns a JSON object mapping each input slide to one of {KEEP, REDUCE, SPLIT} with a one-sentence reason. The prompt enforces Algorithm 1 verbally: it lists the per-slide caps from §A.1, instructs the model that SPLIT is mandatory once images exceed the cap, and forbids the model from emitting slide content at this stage. A short illustrative excerpt:

```
You are an expert presentation content curator.
For each crowded slide, choose ONE action:
1. SPLIT  - divide into N consecutive slides.
2. REDUCE - drop less-critical bullets or
            paraphrase long paragraphs.
3. KEEP   - accept the density.

Decision matrix:
- effective_images > {max_images} -> SPLIT
- effective_images <= 2 BUT bullets
  > {max_bullets_with_imgs} -> REDUCE
- chars > {max_text_chars} -> REDUCE
- Cover / TOC / SectionHeader -> KEEP

Output: JSON {"actions": [{idx, action, reason}]}
```

Inspector Execute Prompt. The Execute stage receives the chosen action and the original slide content, and returns one or more revised slide dictionaries that respect the per-slide caps. The prompt includes one of three action-specific rule blocks (SPLIT, REDUCE, KEEP), enforces schema preservation, and prohibits the invention of new image/table paths. Hard constraints: SPLIT must distribute image/table paths across child slides without duplication or loss; REDUCE must preserve the first 1–2 and last bullets and may drop at most 50% of the source characters; KEEP echoes the input verbatim.

SOF advisory format. Between batches of slide-level tool calls, the executor injects a JSON-formatted user message containing the current occupied regions, up to three suggested free areas on a 25-cell scale (0–24), and a hint to use those free areas. The advisory is informational only and is not enforced. Residual violations are resolved by the Pre-render Rule (§A.3).

A.5 Anchor Mappings for P/D Measurement

To compute PtD-Loss in a system-agnostic way, we anchor P (Planned visual count) to each system's final symbolic-planning artifact: the last point where the system commits to a specific count of visual references before rendering. D (Delivered) counts visually delivered elements in the final PPTX. By default this corresponds to picture shapes. For systems that render tables as native table objects rather

than as embedded images (e.g., PPTAgent), table shapes are also counted, so that tables—which all three systems plan as visual references—are treated consistently across rendering primitives.

System	P anchor	Source artifact
SLOT	post-Inspector slide schema	Inspector output (post-Plan/Execute slide content)
PPTAgent	outline phase	planning-stage outline
SlideTailor	layout-matching phase	layout-matching outline

For SLOT we additionally measure D-unique (count of distinct SHA1-hashed image blobs in the final PPTX) to account for SLOT's cross-slide multi-use: the same source image may appear on multiple slides intentionally. PtD-Loss is reported on the D-unique basis to avoid over-crediting multi-use. The same uniqueness normalization is applied consistently to baseline systems, but since neither PPTAgent nor SlideTailor reuses images across slides in our PSP runs, their D-unique values coincide with their raw D values; the reported baseline figures are therefore unchanged by this normalization.

B Overlap-Severity Protocol

The Overlap-Severity protocol quantifies image-pair occupancy violations in the final rendered PPTX. It is reference-free and judge-free, parallel to PtD-Loss but orthogonal in axis (spatial quality, not delivery fidelity).

Bounding-box extraction.

```
SW = float(pres.slide_width)
SH = float(pres.slide_height)
for shape in slide.shapes:
    if shape.shape_type == 13:  # PICTURE
        bbox = (shape.left/SW,
                shape.top/SH,
                shape.width/SW,
                shape.height/SH)
```

All bounding boxes are axis-aligned (rotated pictures are approximated by their axis-aligned bounding box, AABB) and normalized to $[0, 1]$.

IoU computation. We compute standard axis-aligned IoU for every image pair within the same slide. The aggregate Ov. metric reported in §5.3 (occupancy violations) counts pairs with IoU > 0.005 (any measurable overlap).

Severity buckets.

Bucket	IoU range	Interpretation
Mild	$(0.01, 0.05]$	brushing edges; usually salvageable
Moderate	$(0.05, 0.10]$	partial occlusion
Severe	$(0.10, 0.30]$	one image substantially blocks another
Catastrophic	> 0.30	one image largely on top of another

Scope and limitations. The protocol measures image-image pair overlaps only; text-image and text-text overlaps are excluded because text always lives inside template placeholders (the LLM does not freely allocate text coordinates) whereas images are instantiated with explicit normalized coordinates. Multi-way overlap (3+ images in one region) is counted pairwise and may over-count.

C PSP Dataset Details

The PSP evaluation set is taken from SlideTailor (Zeng et al., 2026). Each entry is a triple (target paper, template, sample reference pair):

```
{
  "target": "6.pdf",
  "template": "8.pptx",
  "sample": {"paper": "27.pdf", "ppt": "27.pdf"}
}
```

The 50 entries cover 50 unique target papers (IDs in $\{6, 7, 9, \ldots, 149\}$) and 9 unique templates ($\{1, 2, 3, 4, 5, 7, 8, 9, 10\}$.pptx). The `sample` field provides an in-context demonstration pair (paper PDF + corresponding deck PPTX) that SlideTailor uses as a few-shot reference; SLOT and PPTAgent do not use this field.

Per-paper statistics (`gpt-4.1-mini`, N=50). SLOT delivers 1,021 unique images total across 50 papers (mean 20.4 / paper, median 19, range 7–64). The Inspector triggers on average 0–2 SPLIT actions per paper (§A.1). Source papers are sampled from arXiv preprints across multiple ML / NLP venues; we did not stratify by venue.

The complete list of pair identifiers ({target}__{template} format) used throughout this paper:

```
6__8, 7__3, 9__8, 10__10, 14__2, 16__7,
18__9, 19__3, 22__9, 24__5, 26__10, 31__1,
33__10, 35__4, 39__4, 46__9, 48__9, 49__2,
51__9, 56__1, 57__8, 58__9, 59__2, 69__7,
70__7, 73__5, 79__8, 85__2, 88__7, 89__4,
93__5, 94__4, 99__8, 100__4, 101__7, 102__5,
106__8, 107__1, 112__8, 113__8, 118__2,
120__7, 124__3, 126__8, 128__8, 130__2,
132__1, 133__8, 146__10, 149__8
```

C.1 Per-template Breakdown

The 50 PSP paper–template pairs are distributed across 9 unique templates with varying placeholder structures. The table below joins (i) the number of pairs using each template, (ii) the per-template PICTURE-placeholder counts (extracted from the template's slide-master layouts), and (iii) aggregate

PtD-Loss across all pairs using that template under `gpt-4.1-mini`.

					PtD-Loss ↓		
Tpl	**#pairs**	**#lyt**	**#PIC**	**0/1/2+ pic**	**PPTAg.**	**SldTlr.**	**SLOT**
1	4	11	1	10/1/0	56.0%	85.3%	**0.0%**
2	6	11	1	10/1/0	60.9%	28.6%	**0.0%**
3	3	4	1	3/1/0	80.4%	77.8%	**0.0%**
4	5	13	1	12/1/0	47.5%	35.1%	**0.0%**
5	4	11	0	11/0/0	61.1%	26.9%	**0.0%**
7	6	15	3	12/3/0	33.3%	97.6%	**0.0%**
8	12	13	3	10/3/0	72.8%	78.8%	**0.0%**
9	6	13	6	7/6/0	70.0%	60.9%	**0.0%**
10	4	12	6	11/0/1	65.0%	53.8%	**0.0%**

Table 5: Per-template breakdown on PSP under `gpt-4.1-mini`. #lyt: number of layouts in the template's slide master; #PIC: total PICTURE placeholders across all layouts; 0/1/2+ pic: distribution of layouts by PICTURE placeholder count. PtD-Loss columns are aggregated across all pairs using that template.

Observations. *(1) Placeholder availability is sparse.* Across the 103 layouts in the 9 templates, only 17 (16.5%) contain any PICTURE placeholder; of those 17, 16 (94%) hold exactly one, a single-image-per-layout schema. Only one layout (in Template 10) provides multiple picture slots. This sparse placeholder structure may partially explain the plan–delivery gap discussed in §3.3: baselines that bind images strictly to placeholder positions can deliver at most approximately 22 images per deck before exhausting the available slots, regardless of how many images the planner commits.

(2) Template-induced variance for baselines. Baseline PtD-Loss varies substantially across templates (PPTAgent 33–80%, SlideTailor 27–98%), suggesting that baseline delivery fidelity is tightly coupled to the specific placeholder structure of the assigned template. Template 5, which has zero picture placeholders, still admits some image delivery via fallback insertion paths but produces variable outcomes across the four pairs that use it.

(3) SLOT is template-invariant. SLOT achieves $\mathcal{L} = 0.0\%$ on every template (computed on the D-unique basis; see §A.5). Because SLOT treats templates as *flexible* and dynamically expands the slide population via SPLIT when capacity is exceeded, the per-template placeholder count does not bottleneck delivery. Inspector SPLIT actions distribute images across newly generated child slides whose layout the Pre-render Rule normalizes geometrically (§A.3).

The aggregate per-template and per-layout statistics needed to support the analysis are reported in Table 5.

D Inspector Branching Distribution

The table summarizes aggregate Inspector action statistics across the 50 PSP papers.

Action	Total	Median/paper	Papers triggered
KEEP	531	9	50/50 (100%)
REDUCE	164	3	46/50 (92%)
SPLIT	84	1	36/50 (72%)

E Per-paper PtD-Loss Distribution

To address the concern that the aggregate PtD-Loss numbers in Table 3 may be driven by outlier papers, we report the per-paper distribution of PtD-Loss across all 50 PSP papers under `gpt-4.1-mini`. The distribution is tabulated below.

System	N	Med.	Mean	p25	p75	Max
PPTAgent	30	64.7%	63.6%	55.6%	77.8%	87.2%
SlideTailor	50	63.6%	62.6%	44.4%	100%	100%
SLOT	40	**0.0%**	**2.6%**	**0.0%**	**0.0%**	35.7%

Observations. SLOT's distribution is tightly concentrated near zero (median 0%, IQR $[0, 0]$), with a small number of outliers reaching up to approximately 36%. Both baselines have median loss $> 60\%$ with substantially wider spread; SlideTailor in particular has 100% loss on multiple decks (papers where no images survive rendering at all). The aggregate Table 3 numbers therefore reflect the central tendency of the per-paper distribution, not a few high-loss outliers inflating the mean.

Sample-size notes. PPTAgent $N{=}30$ because 20 papers lack a parseable planning artifact under `gpt-4.1-mini`; under `gpt-4o`, the same 20 cases fail plus one additional (21/50), indicating that the failure is systematic across backbones rather than stochastic. These unparseable cases are not concentrated in any single template. Main-table aggregates (Tables 1, 3) include all 50 papers, since outline parsing affects per-paper distribution analysis but not the aggregate P, D, and PtD-Loss computation. SlideTailor $N{=}50$ (all outlines available); SLOT $N{=}40$ because 10 papers have inspected slide content with zero image references (cover/TOC-only) for which the loss ratio is undefined.