DIFFBENCH: HOW WELL DO MLLMS UNDERSTAND VISUAL DIFFERENCES?

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-to-code tasks aim to reconstruct visual inputs into editable and executable artifacts with high visual fidelity. During reconstruction, visual difference detection is a key tool for locating reconstruction errors and guiding subsequent refinement. It makes evaluation of visual difference detection increasingly important. Existing difference detection benchmarks largely focus on synthetic artifacts or object-centric natural images, leaving complex documents in real-world underexplored. To address this gap, we introduce DiffBench, a high-quality benchmark of visual differences detection, featuring non-synthetic discrepancies in diverse real-world document types such as slides, magazines and reports. Besides, we propose an automated evaluation framework for assessing visual difference predictions across four dimensions including layout, relation, content integrity and visual properties. Our evaluation reveals that GPT-6 Astra achieves the strongest overall performance and is better than GPT-5.6 Sol, while Kimi K3 leads among open-weight models including its previous version Kimi K2.5. Across models, content integrity discrepancies like missing text and images are generally easier to detect, whereas precise alignment, visual properties, and multi-region layout changes remain substantially more challenging. Performance further drops on highly similar cases with subtle differences, where models increasingly miss true discrepancies. Overall, DiffBench remains challenging for current MLLMs while effectively capturing progress across successive model generations.

1 INTRODUCTION

The vision-to-code task aims to reconstruct visual inputs into editable and executable contents such as webpages (Li et al., 2026), documents (Zhu et al., 2026), and others (Lin et al., 2026; Zhao et al., 2026). In agentic reconstruction, models typically alternate between code generation, rendering, visual inspection, and iterative revision. Accurately identifying differences between the rendered output and the given reference is therefore essential for locating reconstruction errors and providing actionable feedback for subsequent edits. As the effectiveness of this refinement loop depends directly on the quality of such visual feedback, systematically evaluating the capability of visual difference detection becomes essential for understanding and improving agentic reconstruction.

Recent benchmarks increasingly evaluate visual difference detection in MLLMs (Liu et al., 2025; Wei et al., 2026; Zhang et al., 2026b). Existing benchmarks primarily focus on the natural images (Liu et al., 2025; Wei et al., 2026) and synthesized artifacts like HTML/CSS pages (Zhang et al., 2026b). For example, OmniDiff (Liu et al., 2025) evaluates fine-grained image differences across diverse natural and 3D-rendered images, covering a broad range of visual change types, while DiffSpot (Zhang et al., 2026b) constructs HTML-rendered image pairs by modifying a single property of an element. Complementary to these settings, complex real-world documents represent a practically important yet underexplored scenario, where high-fidelity reconstruction into editable artifacts requires models to detect document-specific discrepancies in layout, alignment, and typography, as shown in Figure 1.

To this end, we introduce DiffBench, a high-quality benchmark of non-synthetic visual differences detection in real-world document scenario. Specifically, DiffBench constructs 578 diverse image pairs spanning presentation slides, editorial publications, institutional documents, commercial materials, and technical documents (as shown in Figure 1). And we curate 12,267 verified visual

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

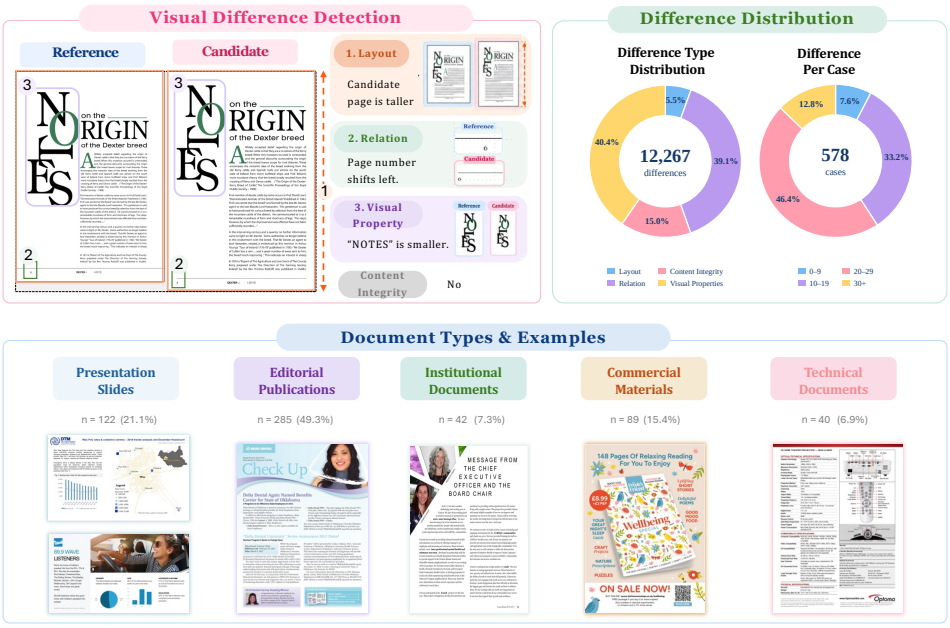

Figure 1: **Overview of DiffBench. Top left:** The visualization of the visual difference detection and the evaluation across four difference types on this task. **Top right:** Difference distribution. **Bottom:** Diverse real-world document types and examples in DiffBench.

differences with high quality including easy to hard. Unlike benchmarks based on manually injected or isolated perturbations, our candidate images are produced through vision-to-code reconstruction, preserving the naturally co-occurring errors encountered in practical scenarios. To obtain high-quality annotations at scale, we develop an agentic MLLM reasoning construction pipeline that establishes localized element references, performs hierarchical comparison across page, region, and element levels, and consolidates independently discovered differences through multi-model verification.

In addition, evaluating model predictions in this setting is also challenging, as there is no established evaluation framework tailored to this scenario. Given the inherently hierarchical structure of documents, visual differences may arise from global layout and cross-element relations to local content and appearance. We therefore define four complementary evaluation dimensions, *Layout*, *Relation*, *Content Integrity*, and *Visual Properties*, together with corresponding sub-dimensions, and further label each difference by its criticality as *Major*, *Medium*, or *Minor*, enabling comprehensive analysis of both model capability. We then develop an automated evaluation framework that assigns these dimensions to both reference and predicted differences, and then performs dimension-conditioned semantic matching based on the affected visual target and the underlying visual change. It further separately evaluates the degree of coverage on each dimension for matched difference, distinguishing fully correct, partially correct, and erroneous descriptions. Based on these judgments, we report accuracy on each dimension and reveal what type of difference each model can reliably detect, where it struggles, and if frontier MLLMs can capture fine-grained differences.

Our contributions can be summarized as follows:

- We introduce DiffBench, a high-quality benchmark for visual difference detection across diverse real-world document domains, which will serve as a public testbed for evaluating and advancing agents in this setting.
- We develop a structured evaluation protocol and automated framework for assessing free-form visual difference detection across four difference dimensions, revealing model strengths and failure modes.
- Our findings show that most MLLMs still struggle with complex relations, visual properties, multi-region layout changes, and subtle differences, while consistent improvements across successive models within the same family (GPT-6 Astra vs 5.6 sol, Kimi K3 vs Kimi K2.5) demonstrate DiffBench's ability to track model progress.

2 Related Work

Vision-to-Code Task. Recent vision-to-code research increasingly uses MLLMs to reconstruct visual interfaces into executable code (Liang et al., 2025; Awal et al., 2025; Si et al., 2025; Chen et al., 2025; Deng et al., 2026; Li et al., 2026). Early work mainly adopts a single-pass paradigm that directly translates screenshots or UI designs into HTML/CSS or other executable representations. WAFFLE (Liang et al., 2025), for example, trains MLLMs to better model HTML structure and fine-grained discrepancies between reference UIs and rendered outputs, improving reconstruction fidelity. Design2Code (Si et al., 2025) manually curates 484 webpages to evaluate frontier MLLMs for visual-to-code reconstruction, revealing persistent challenges in visual element recall and layout fidelity. More recent work shifts toward iterative and agentic visual coding (Li et al., 2026; Chen et al., 2025; Deng et al., 2026), where agents repeatedly inspect rendered outputs and refine code using visual feedback. ReLook (Li et al., 2026), for instance, trains a coding agent with iterative visual refinement guided by an MLLM critic, while related methods use rendering-based feedback or explicit discrepancy signals for self-correction (Chen et al., 2025; Deng et al., 2026). As this paradigm becomes increasingly important, accurately identifying fine-grained discrepancies between references and rendered outputs serves not only as an evaluation signal, but also as feedback or supervision for improving generation.

Visual Difference Benchmarks. Visual difference benchmarks (Jhamtani & Berg-Kirkpatrick, 2018; Forbes et al., 2019; Liu et al., 2025; Wei et al., 2026; Zhang et al., 2026b) span a variety of settings, most commonly natural images and synthetically constructed visual artifacts. For natural images, prior work mainly studies object- and scene-level differences involving appearance, attributes, presence, and spatial changes (Jhamtani & Berg-Kirkpatrick, 2018; Forbes et al., 2019; Liu et al., 2025; Wei et al., 2026; Zhang et al., 2026a). Another line of work considers code-rendered artifacts, where visual differences are introduced through controlled perturbations (Zhang et al., 2026b). For example, DiffSpot (Zhang et al., 2026b) constructs Web UI pairs by mutating a single CSS property of a target element, while WUICC-bench (Zhang et al., 2026a) applies controlled edits to content, layout, resizing, reordering, and styling. In contrast, DiffBench focuses on complex real-world documents and naturally occurring reconstruction errors, where multiple fine-grained discrepancies may co-occur across layout, spatial relations, content, and visual appearance.

3 DiffBench: Design and Construction

DiffBench evaluates fine-grained visual difference detection in complex real-world documents across layout, relations, content integrity, and visual properties. The following sections describe image-pair collection and golden-difference construction.

3.1 Image Pair Collection

Each example in DiffBench consists of a reference image and a corresponding candidate image. The reference defines the target visual appearance, while the candidate to be compared is produced through MLLM-based reconstruction. We collect reference images from publicly available PDF documents and render their pages into images, spanning five domains: presentation slides, editorial publications, commercial materials, technical documents, and institutional documents. To curate visually rich and challenging examples, we use an MLLM-as-a-judge to assess each image based on document type, layout structure, typography, color composition, visual assets, information and visual density, reading-order complexity, reconstruction difficulty, and contribution to dataset diversity (more details are shown in Appendix B.1). We retain high-scoring candidates and manually review them to ensure visual quality and diversity.

To construct candidate images, we use multiple frontier MLLMs to perform image-to-editable reconstruction for each reference image, producing naturally occurring reconstruction errors rather than manually injected perturbations (Appendix B.2). The generated HTML is rendered in a consistent browser environment, after which we manually remove failed reconstructions and pairs with insufficient visual discrepancy. We retain visually similar reference-candidate pairs with meaningful reconstruction-induced differences, including 46 hard cases with fewer and subtler discrepancies for evaluating fine-grained visual sensitivity.

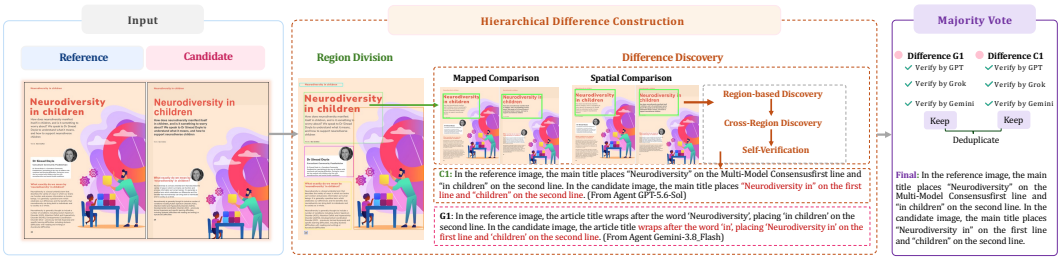

Figure 2: **Overview of the DiffBench difference construction pipeline.** Given a reference-candidate image pair, we first discover all visible elements of the reference image and then multiple agents then perform hierarchical difference discovery through region-based inspection, cross-region reasoning, and fresh-context self-verification. The resulting difference candidates are independently verified by multiple MLLMs, retained through majority voting, and semantically deduplicated to produce the final golden differences.

3.2 Difference Construction

To construct comprehensive and reliable golden differences with high quality, we adopt a multi-stage, multi-model annotation pipeline (shown in Figure 2). The pipeline consists of multi-stages: 1) element reference discovery, where multiple MLLMs independently identify and consolidate visible elements; 2) hierarchical difference discovery; 3) multi-Model Consensus and Consolidation, we use heterogeneous MLLMs as independent annotators whenever possible to improve coverage and reduce model-specific bias.

Element Reference Discovery. To support consistent multi-model annotation and fine-grained difference discovery, we first establish a shared element-level reference for each image. For each reference image, multiple MLLMs independently identify all the concrete and referable visual elements, including text blocks, images, illustrations, charts, icons, shapes, and fine-grained layout components such as dividers, rules, borders, and connectors. The outputs from these MLLMs are then semantically merged by a separate MLLM, which clusters mentions of the same physical element, removes duplicates, and assigns each cluster a canonical description. The resulting elements are finally localized on the reference image to obtain bounding boxes, which enable human verification of element quality and provide spatial context for subsequent difference discovery. More details of prompts can be found in Appendix B.3.

Hierarchical Difference Construction. Visual difference detection should consider hierarchical visual structures rather than treating discrepancies as isolated element-level changes. Humans intuitionally often compare documents by first perceiving global composition, then inspecting local regions with spatial relationship and individual elements. Similarly, a single visual modification may propagate across multiple related elements within or across regions, resulting in discrepancies at different granularities. For example, enlarging a title may change its own appearance, alter text wrapping, shift surrounding elements, and eventually introduce overlap or clipping. Treating these consequences as independent differences can lead to redundant cascaded descriptions, while element-level comparison alone may miss the higher-level visual change.

Motivated by the inherent layouts of documents, we develop an agentic hierarchical difference discovery skill that leverages the nature of regions. To be more specific, Complex documents naturally organize individual elements into coherent regions through spatial relationships, alignment, and reading order. Regions serve as an intermediate level between the whole page and individual elements, preserving contextual information while enabling fine-grained discrepancy localization. Moreover, region-level reasoning allows the agent to capture coherent visual changes at an appropriate granularity and reduce redundant cascaded differences caused by the same visual modification.

The agentic skill consists of three components. First, 1) **region division and mapping**: The agent constructs semantic regions from discovered elements, layout structures, alignment relationships, and reading order, and maps each region to its semantic counterpart in the candidate image. For each region, it generates two complementary comparison views: spatial comparison, where the reference region is placed at the same coordinates in both images to reveal changes in position, scale,

and layout; and mapped comparison, where corresponding regions are aligned to compare content, appearance, and internal structures. 2) **region-based difference discovery**: based on these region-level visual representations, the agent inspects local regions, interactions among regions, and the overall page context to identify discrepancies at appropriate granularities. Instead of decomposing a single visual change into multiple downstream effects, the agent describes differences at the highest visually supported level. 3) **Fresh-context self-verification**: the agent revisits the original visual evidence and generated differences in a fresh context to refine the results by removing unsupported claims, recovering missed discrepancies, correcting inaccurate descriptions, and consolidating duplicate or cascaded findings.

As aforementioned, to further improve coverage and reduce individual model bias, we instantiate multiple agents based on the above hierarchical difference discovery skill. Each agent independently inspects the image pair and generates difference candidates from its own visual reasoning process (more details in Appendix B.4).

Multi-Model Consensus and Consolidation. The generated difference candidates are then independently evaluated by multiple MLLMs. Each verifier determines whether a proposed difference is visually supported and should be retained. We use majority voting to filter unreliable candidates, keeping differences supported by more than half of the verifiers. Finally, a dedicated MLLM performs semantic deduplication and consolidation to merge equivalent descriptions and produce the final set of differences (details in Appendix B.5).

4 EXPERIMENTS

In this section, we will include the detailed setup of our experiments, the evaluation protocol and data statistics of DiffBench.

4.1 EXPERIMENT SETUP

We use different frontier agents across the DiffBench pipeline. Specifically, for diverse candidate image construction, we employ Gemini-3.1-Pro-Preview (Google DeepMind, 2026a), GPT-5.5 (OpenAI, 2026a), GPT-5.6 Sol (OpenAI, 2026b), Grok-4.3 (SpaceXAI, 2026a), and Qwen3.5-122B-A10B (Qwen Team, 2026a) to independently perform image-to-code reconstruction. For element reference discovery in Section 3.2, GPT-5.6 Sol (OpenAI, 2026b), Gemini-3.8-Flash (Google DeepMind, 2026b), and Grok-4.6 (SpaceXAI, 2026b) independently identify visible elements, while Gemini-3.8-Flash (Google DeepMind, 2026b) is further used for merging and element localization due to strong grounding capability. For hierarchical difference discovery, we independently instantiate three agents including GPT-5.6 Sol (OpenAI, 2026b), Gemini-3.8-Flash (Google DeepMind, 2026b), and Grok-4.6 (SpaceXAI, 2026b) in agentic mode with high reasoning effort and the same skill. The resulting difference candidates are then verified through the same three models and use GPT-5.6 sol (OpenAI, 2026b) to summarize the final golden differences. Details for each stage are provided in Appendix B.

4.2 EVALUATION PROTOCOL

In this section, we present our evaluation protocol for free-form visual difference predictions. It consists of evaluation dimension and an automatic evaluation framework.

4.2.1 EVALUATION DIMENSIONS.

To characterize what aspects of a document are affected by each visual discrepancy, we organize the evaluation into four dimensions, each further decomposed into several sub-dimensions (more detailed definition of each dimension in Appendix Table 2). Specifically, 1) ***Layout*** captures differences in the global or page-level structure and composition, including *page geometry*, *layout skeleton*, and *global composition*; 2) ***Relation*** evaluates spatial relationships among elements or between an element and its surrounding boundary through *position*, *alignment*, *spacing*, *topology*, and *flow*; 3) ***Content Integrity*** measures whether semantic content is correctly presented, covering *text & data*, *visual content like image or illustration*, *symbols & graphics like icons, logos* and *structured content such as charts, tables, diagrams* ; 4) ***Visual Properties*** assesses the intrinsic appearance

Model	Overall		Layout		Relation		Content Integrity		Visual Property	
	Weighted ↑	Strict ↑	Weighted ↑	Strict ↑	Weighted ↑	Strict ↑	Weighted ↑	Strict ↑	Weighted ↑	Strict ↑
Proprietary Models										
GPT-6 Astra	**64.75**	**55.44**	**42.22**	**35.61**	**65.35**	**54.63**	**72.31**	**65.67**	**64.43**	**55.11**
GPT-5.6-Sol	52.18	41.80	41.66	33.59	55.46	42.24	60.36	54.12	47.39	37.90
Δ vs. GPT-5.6-Sol	+12.57	+13.64	+0.56	+2.02	+9.89	+12.39	+11.95	+11.55	+17.04	+17.21
Grok 4.6	20.18	14.99	17.12	14.21	14.76	8.51	59.62	52.72	11.16	7.31
Gemini 3.8 Flash	19.16	14.31	34.15	31.68	14.83	8.80	51.82	44.98	9.14	5.85
Open-Weight Models										
Kimi K3	24.83†	21.12†	12.88†	8.85†	32.49†	29.91†	48.30	40.88	10.30†	6.93†
Kimi K2.5	12.70	8.68	3.36	1.12	8.14	3.96	42.12	34.54	7.43	4.66
Δ vs. Kimi K2.5	+12.13	+12.44	+9.52	+7.73	+24.35	+25.95	+6.18	+6.34	+2.87	+2.27
Qwen3.8-27B	21.19	15.12	11.23	6.49	17.45	10.38	48.71†	41.04†	15.91†	11.21†
DeepSeek V4 Pro	0.07	0.02	0.11	0.00	0.04	0.00	0.07	0.00	0.11	0.06

Table 1: **Visual difference detection performance across representative models.** Weighted denotes the partial-credit score; Strict awards credit only for fully correct coverage without same-target errors. All scores are percentages. Blue shading and boldface indicate the best result overall; underlining with † indicates the best result among open-weight models with available results. Green Δ rows show within-family changes relative to the named baseline, in percentage points. Dashes indicate unavailable results.

and geometry of individual elements, including *color & appearance*, *typography like font family, font size*, *size & shape*, and *border & effects such as borders, strokes and corner radius*. It enables evaluation at both the category and subcategory levels, providing a more diagnostic view of model strengths and failure modes.

4.2.2 Evaluation Framework

Dimension Assignment. We first assign the above difference-type annotations to both golden differences and model predictions using an MLLM-based annotator. Since a visual difference may describe multiple related aspects and is not necessarily atomic, each difference can receive multiple type tags. For every assigned tag, the annotator additionally provides a short rationale indicating which part of the difference supports that annotation. We separately annotate the criticality of each golden difference including *Major*, *Medium* and *Minor*. More details can be seen in Appendix B.6.

Dimension-Conditioned Matching and Scoring. Given these annotations, we group golden differences and predictions by type dimension while retaining each original difference description in its entirety. We avoid extracting dimension-specific fragments, since rewriting or decomposition may introduce additional noise. Within each dimension, a prediction is matched to a golden difference when they refer to the same visual target and event. The evaluator (MLLM-as-a-judge) receives the raw differences together with their tags and rationales, and is instructed to consider only the content relevant to the current dimension.

Dimension-aware Accuracy. For each golden difference g associated with dimension d, we assign a coverage score $s(g, d)$ based on the evaluator's judgment. Specifically, *full coverage without error* receives 1.0, *full coverage with error* receives 0.75, *partial coverage without error* receives 0.5, *partial coverage with error* receives 0.25, and *no valid coverage* receives 0. We first report **Strict Accuracy**, which only counts golden differences that are fully and correctly covered:

$$\mathrm{StrictAcc}_d = \frac{1}{|G_d|} \sum_{g \in G_d} \mathbb{I}\big[s(g, d) = 1\big], \tag{1}$$

where G_d denotes the set of golden differences associated with dimension d. We further report **Weighted Accuracy**, which gives partial credit according to the coverage quality:

$$\mathrm{WeightedAcc}_d = \frac{1}{|G_d|} \sum_{g \in G_d} s(g, d). \tag{2}$$

4.3 Data Statistics

Consequently, as shown in Figure 1, DiffBench introduces 12,267 diverse and reliable visual differences on 578 image pairs. Specifically, The reference images span five real-world document domains: editorial publications account for 285 cases (49.3%), presentation slides for 122 (21.1%), commercial materials for 89 (15.4%), institutional documents for 42 (7.3%), and technical documents for 40 (6.9%). This distribution reflects the diversity of document structures, visual styles, and information densities covered by the benchmark. The annotated differences consider the diversity of types and the hard level. For the type distribution across four dimensions, there are 40.4% *Visual Properties*, 39.1% *Relation*, 15.0% *Content Integrity* and 5.5% *Layout*. We also include 46 hard cases in which the reference and candidate images are highly similar and differ only in a small number of subtle visual details, enabling targeted evaluation of model performance under more challenging comparison settings. In addition, the number of differences per case also varies substantially: 7.6% of cases contain fewer than 10 differences, 33.2% contain 10-19, 46.4% contain 20-29, and 12.8% contain 30 or more. Together, these statistics highlight both the diversity of source documents and the heterogeneous, naturally co-occurring discrepancies captured by DiffBench. Qualitative examples of DiffBench across diverse document types are provided in Appendix D.

4.4 Model And Human Verification of DiffBench

We assess the reliability of DiffBench through cross-model consistency and human verification. Three heterogeneous MLLMs independently perform element discovery, their agreement reaches 96% and we also use majority vote in the final difference merge stage to avoid bias and unreliability. And we incorporate manual inspection throughout benchmark construction. Among the 12,620 audited difference candidates, 12,267 (97.2%) were fully supported by the visual evidence, while the remaining 353 (2.8%) required revision or removal. Across 500 independently reviewed evaluation on three selected evaluated models' decisions (including GPT-6-Astra (OpenAI, 2026c), Kimi K3 (Kimi Team, 2026) and Grok-4.6 (SpaceXAI, 2026b)), the average of human–evaluator agreement is 88.0% for five-level coverage scores and 94.0% for strict correctness. In a separate sample of 500 predictions from three models, 2, 4 and 5 are visually unsupported, corresponding to 0.4%, 0.8% and 1.0% hallucination rates separately. The above proves the high quality of DiffBench.

5 Results & Analysis

5.1 Main Results

We evaluate the performance based on frontier models including GPT-6 Astra (OpenAI, 2026c), GPT-5.6 sol (OpenAI, 2026b), Gemini-3.8-Flash (Google DeepMind, 2026b), Grok-4.6 (SpaceXAI, 2026b), Qwen-3.8-27B (Qwen Team, 2026b), Kimi-K3 (Kimi Team, 2026), Kimi-K2.5 (Kimi Team et al., 2026) and DeepSeek-V4-Pro (DeepSeek-AI, 2026). As shown in Table 1, GPT-6 Astra (OpenAI, 2026c) achieves the best Weighted and Strict Accuracy across all four dimensions. Compared with GPT-5.6-Sol (OpenAI, 2026b), Astra improves substantially in most dimensions, especially Visual Properties (+17.04 points), however, *Layout* remains challenging for the highest-scoring model with Astra obtaining 42.22% Weighted Accuracy and 35.61% Strict Accuracy. Beyond the GPT family, Kimi also shows clear within-family progress where Kimi K3 (Kimi Team, 2026) improves most on *Relation* (+24.35), but only modestly on *Visual Properties* (+2.87). These consistent improvements across two model families suggest that DiffBench can distinguish performance gains across model generations.

Across models, *Content Integrity* is generally the strongest dimension. Grok 4.6 (SpaceXAI, 2026b), Gemini-3.8-Flash (Google DeepMind, 2026b), and Kimi K3 (Kimi Team, 2026) obtain 48.30˜59.62% Weighted Accuracy on *Content Integrity*, but only 9.14˜11.16% on *Visual Properties*, indicating that strong performance in detecting content presence does not necessarily translate to comparable sensitivity to changes in the visual properties of elements in the images. These three models achieve similar overall Weighted Accuracy (19.16˜24.83%), yet exhibit different strengths: Kimi K3 (Kimi Team, 2026) performs best on *Relation* (32.49%), Grok-4.6 (SpaceXAI, 2026b) on *Content Integrity* (59.62%) and Gemini-3.8-Flash (Google DeepMind, 2026b) on *Layout* (34.15%). Finally, DeepSeek-V4-Pro (DeepSeek-AI, 2026) performs poorly across all dimensions, with only 0.07% overall Weighted Accuracy and 0.02% Strict Accuracy.

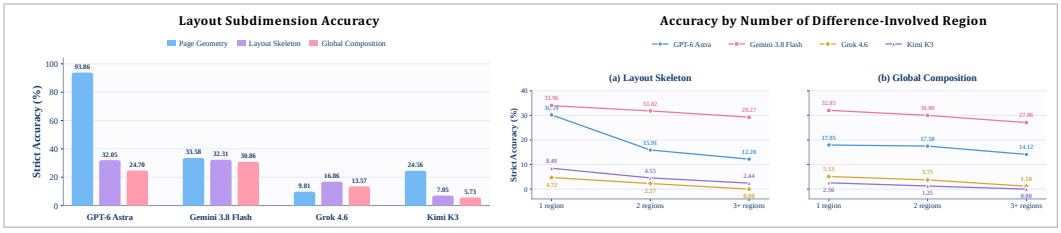

Figure 3: **Sub-dimension analysis of *Layout*. Left:** Strict Accuracy across the three layout sub-dimensions for representative models. **Right:** Strict Accuracy on *Layout Skeleton* and *Global Composition* grouped by the number of regions involved in each difference.

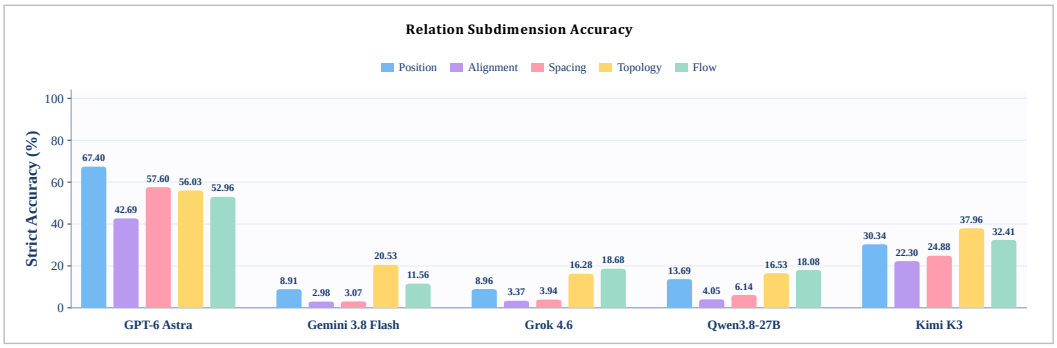

Figure 4: **Sub-dimension analysis of *Relation*.** Strict Accuracy across five relation sub-dimensions for representative models, with *Alignment* consistently showing the lowest performance.

5.2 Deep Analysis & Findings

We further examine model performance at the sub-dimension level and report two import findings on *Layout* and *Relation* (more sub-dimension analysis around other dimensions are shown in Appendix C.1), hard cases analysis and error analysis on Deepseek-V4-pro (DeepSeek-AI, 2026).

5.2.1 Sub-dimension Analysis

Layout. In terms of *Layout* (as shown in Figure 3), ***performance on Layout is largely driven by Page Geometry, while Layout Skeleton and Global Composition remain substantially harder.*** This is especially evident for GPT-6 Astra (OpenAI, 2026c), which achieves 93.86% Strict Accuracy on *Page Geometry*, but only 32.05% and 24.70% on the other two sub-dimensions. We attribute this gap to their different spatial scopes: page geometry mainly involves direct page-level properties such as dimensions and aspect ratio, and can often be determined through relatively direct comparisons of the page as a whole, whereas layout skeleton and global composition require integrating relationships across multiple regions.

To further test this hypothesis, we group *Layout Skeleton* and *Global Composition* differences by the number of involved regions using the region information retained during construction. Strict Accuracy generally decreases as more regions are involved, with a particularly clear drop for GPT-6 Astra (OpenAI, 2026c), suggesting that multi-region integration is a major source of difficulty in layout difference detection. Gemini-3.8-Flash (Google DeepMind, 2026b), in contrast, shows relatively balanced performance across layout sub-dimensions. Overall, these results suggest that the difficulty of layout difference detection increases with the number of regions, especially in layout skeleton and global composition detection.

Relation. As shown in Figure 4, ***Alignment* is consistently the most challenging relation sub-dimension across models.** Alignment requires precise comparison of shared edges, centers, axes, or baselines across elements, making it particularly sensitive to subtle spatial deviations. Notably, this bottleneck persists even for GPT-6 Astra (OpenAI, 2026c) despite its substantial gains on other relation types. For most other models, *Topology* and *Flow* are considerably easier than alignment,

suggesting that models are better at recognizing more salient structural relationships, such as containment, overlap, or reflow, than precise geometric correspondence between elements.

5.2.2 Hard Cases Analysis: Fewer but More Difficult Differences

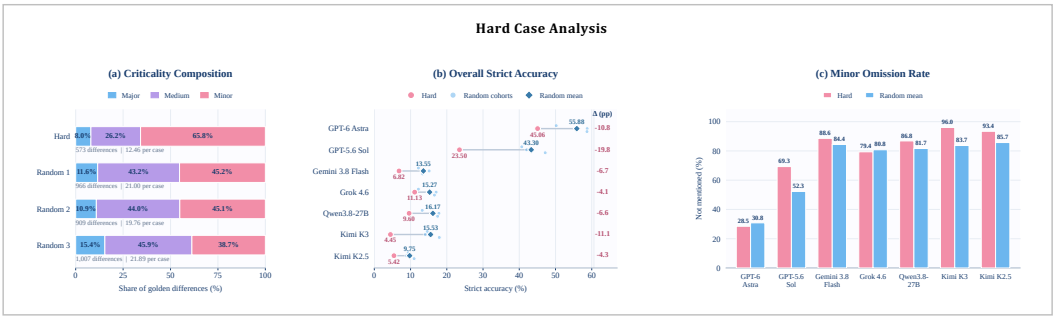

Figure 5: **Hard-case analysis.** Criticality composition, Strict Accuracy, and Minor omission rates on hard cases versus random controls.

We further investigate model performance on hard cases curated in our benchmark, where the reference and candidate images are highly similar and differ only in a small number of subtle visual details. To examine whether current models remain reliable under such low-salience comparison settings, we fairly compare the 46 hard cases against three non-overlapping random subsets containing the same number of cases.

As shown in Figure 5, the hard subset contains substantially fewer differences per case and a markedly higher proportion of *Minor* differences than random controls. Correspondingly, all effective models achieve lower Strict Accuracy on the hard cases than on the random subsets, with particularly large drops for GPT-5.6 sol (OpenAI, 2026b), Kimi-K3 (Kimi Team, 2026), and GPT-6 Astra (OpenAI, 2026c). Moreover, the omission rate of *Minor* differences generally increases on the hard subset, suggesting that the performance degradation primarily reflects reduced sensitivity to subtle discrepancies rather than an artifact of evaluation-set size. Additional element-level analysis in Appendix C.2 also highlights limited coverage of fine-grained visual changes. ***Overall, these results indicate that most models still remain dependent on visually salient cues and struggle to detect sparse differences between highly similar document images.***

5.3 Error Analysis

As shown in Table 1, DeepSeek-V4-Pro (DeepSeek-AI, 2026) performs poorly across four dimensions. We further analyze the failure mode from predictions, the format of uts prediction is correct, and among 16,231 ground-truth dimension units, only four are fully correct (0.025%), whereas 99.71% are classified as Not Mentioned where no prediction substantively covers the target Gold difference. Partial and Incorrect/Error outcomes account for only 0.13% and 0.14%, respectively. The primary bottleneck is identifying and grounding the correct visual difference events, rather than describing correctly detected events incompletely. ***We believe it is due to the actual failure in visual difference perception.***

6 Conclusion

We introduce DiffBench, a benchmark for visual difference detection in complex real-world documents, comprising 578 image pairs with five document types and 12,267 verified hierarchical differences. On top of data construction, we further propose a structured evaluation protocol that assesses free-form predictions across four difference dimensions and criticality levels, enabling fine-grained diagnosis of model strengths, failure modes, and the capability of fine-grained visual discrepancies detection. Experiments reveal that frontier MLLMs generally detect content discrepancies more successfully than precise alignment, visual property changes, and multi-region layout differences. Performance further declines on highly similar image pairs containing sparse, subtle discrepancies, highlighting persistent challenges in complete and accurate visual comparison.

AI USE STATEMENT

Generative AI tools were extensively used in the construction and evaluation of DiffBench. In particular, frontier multimodal language models were used for reference-page characterization and selection, vision-to-code reconstruction of candidate images, element discovery and localization, hierarchical visual difference discovery, multi-model verification and consolidation, difference dimension annotation, and automated evaluation of model predictions. Multiple models were used independently at several stages to improve coverage and reduce dependence on a single model.

AI-generated annotations and intermediate results were also verified by manual inspection. Reference images were manually reviewed after model-based selection, localized elements were inspected for annotation quality, and candidate differences were subjected to independent multi-model verification and majority voting before consolidation. We additionally performed manual inspection throughout benchmark construction and analysis to identify invalid, redundant, or unsupported outputs. All reported experimental results and conclusions were reviewed by the authors.

Generative AI tools were also used to assist with implementation, prompt development, and language editing during the preparation of this work. All AI-assisted code, prompts, analysis, and manuscript text were reviewed and modified by the authors as needed. We take responsibility for the final content of this work, including all text, claims, annotations, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

DiffBench is designed to evaluate visual difference detection in real-world document. The benchmark is constructed from publicly accessible document materials and model-generated reconstructions, and does not involve human-subject experiments. We manually review selected samples and do not intentionally collect private or sensitive information. For dataset release, we will respect the licenses and redistribution conditions of the original source materials and release derived annotations or references to source content where direct redistribution is not permitted.

Because multi-modal models are used extensively during benchmark construction, their errors or biases may propagate into intermediate annotations. We mitigate this risk through independent multi-model generation, verification, consensus filtering, and manual inspection. Nevertheless, the benchmark may still contain annotation errors or reflect biases present in the underlying models and source documents. DiffBench is intended for research on visual comparison and document reconstruction, and model-generated reconstructions or annotations should not be treated as authoritative representations of the semantic content of the source documents.

REFERENCES

Rabiul Awal, Mahsa Massoud, Aarash Feizi, Zichao Li, Suyuchen Wang, Christopher Pal, Aishwarya Agrawal, David Vázquez, Siva Reddy, Juan A. Rodríguez, Perouz Taslakian, Spandana Gella, and Sai Rajeswar. Webmmu: A benchmark for multimodal multilingual website understanding and code generation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 25118–25145. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.1276. URL https://doi.org/10.18653/v1/2025.emnlp-main.1276.

Yang Chen, Yufan Shen, Wenxuan Huang, Sheng Zhou, Qunshu Lin, Xinyu Cai, Zhi Yu, Jiajun Bu, Botian Shi, and Yu Qiao. Learning only with images: Visual reinforcement learning with reasoning, rendering, and visual feedback. *CoRR*, abs/2507.20766, 2025. doi: 10.48550/ARXIV.2507.20766. URL https://doi.org/10.48550/arXiv.2507.20766.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro. Technical report and model card.

Jie Deng, Kaichun Yao, and Libo Zhang. Visrefiner: Learning from visual differences for screenshot-to-code generation. *CoRR*, abs/2602.05998, 2026. doi: 10.48550/ARXIV.2602.05998. URL https://doi.org/10.48550/arXiv.2602.05998.

Maxwell Forbes, Christine Kaeser-Chen, Piyush Sharma, and Serge J. Belongie. Neural naturalist: Generating fine-grained image comparisons. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 708–717. Association for Computational Linguistics, 2019. doi: 10.18653/V1/D19-1065. URL https://doi.org/10.18653/v1/D19-1065.

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

Google DeepMind. Gemini 3.8 Flash model card, 2026b. URL https://deepmind.google/models/model-cards/gemini-3-8-flash/. Model card.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. Learning to describe differences between pairs of similar images. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 4024–4034. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1436. URL https://doi.org/10.18653/v1/d18-1436.

Kimi Team. Kimi K3: Open frontier intelligence, 2026. URL https://github.com/MoonshotAI/Kimi-K3/blob/main/k3_tech_report.pdf. Technical report.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, et al. Kimi K2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

Yuhang Li, Chenchen Zhang, Ruilin Lv, Ao Liu, Ken Deng, Yuanxing Zhang, Jiaheng Liu, and Bo Zhou. Relook: Vision-grounded RL with a multimodal LLM critic for agentic web coding. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 25471–25485. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.1167. URL https://doi.org/10.18653/v1/2026.acl-long.1167.

Shanchao Liang, Nan Jiang, Shangshu Qian, and Lin Tan. WAFFLE: fine-tuning multi-modal model for automated front-end development. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 24786–24802. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.1208. URL https://doi.org/10.18653/v1/2025.acl-long.1208.

Zhen Lin, Qiujie Xie, Minjun Zhu, Shichen Li, Qiyao Sun, Enhao Gu, Yiran Ding, Ke Sun, Fang Guo, Panzhong Lu, Zhiyuan Ning, Yixuan Weng, and Yue Zhang. Autofigure-edit: Generating editable scientific illustrations via reference-guided styling. In Greg Durrett and Ping Jian (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 57–67. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-DEMO.6. URL https://doi.org/10.18653/v1/2026.acl-demo.6.

Yuan Liu, Saihui Hou, Saijie Hou, Jiabao Du, Shibei Meng, and Yongzhen Huang. Omnidiff: A comprehensive benchmark for fine-grained image difference captioning. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25, 2025*, pp. 21440–21449. IEEE, 2025. doi: 10.1109/ICCV51701.2025.01991. URL https://doi.org/10.1109/ICCV51701.2025.01991.

OpenAI. GPT-5.5 system card, 2026a. URL https://openai.com/index/gpt-5-5-system-card/. System card.

OpenAI. GPT-5.6 system card, 2026b. URL https://deploymentsafety.openai.com/gpt-5-6. System card.

OpenAI. GPT-6 Astra system card, 2026c. URL https://deploymentsafety.openai.com/gpt-6-astra. System card.

Qwen Team. Qwen3.5-122B-A10B model card, 2026a. URL https://huggingface.co/Qwen/Qwen3.5-122B-A10B. Model card.

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

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 (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pp. 3956–3974. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.NAACL-LONG.199. URL https://doi.org/10.18653/v1/2025.naacl-long.199.

SpaceXAI. Grok 4.3, 2026a. URL https://docs.x.ai/developers/models/grok-4.3. Official model documentation.

SpaceXAI. Introducing Grok 4.6, 2026b. URL https://x.ai/news/grok-4-6. Official release announcement.

Yuancheng Wei, Haojie Zhang, Linli Yao, Lei Li, Jiali Chen, Tao Huang, Yiting Lu, Duojun Huang, Xin Li, and Zhao Zhong. Diffcap-bench: A comprehensive, challenging, robust benchmark for image difference captioning. *CoRR*, abs/2605.04503, 2026. doi: 10.48550/ARXIV.2605.04503. URL https://doi.org/10.48550/arXiv.2605.04503.

Licheng Zhang, Bach Le, Pengtao Zhao, and Naveed Akhtar. Beyond pixel diffs: Benchmarking image change captioning for web UI visual regression testing. *CoRR*, abs/2607.01728, 2026a. doi: 10.48550/ARXIV.2607.01728. URL https://doi.org/10.48550/arXiv.2607.01728.

Linhao Zhang, Aiwei Liu, Yuan Liu, and Xiao Zhou. Diffspot: Can vlms spot fine-grained visual differences in web interfaces? *CoRR*, abs/2605.29615, 2026b. doi: 10.48550/ARXIV.2605.29615. URL https://doi.org/10.48550/arXiv.2605.29615.

Haozhe Zhao, Shuzheng Si, Zhenhailong Wang, Zheng Wang, Liang Chen, Xiaotong Li, Zhixiang Liang, Maosong Sun, and Minjia Zhang. Crafter: A multi-agent harness for editable scientific figure generation from diverse inputs. *CoRR*, abs/2605.30611, 2026. doi: 10.48550/ARXIV.2605.30611. URL https://doi.org/10.48550/arXiv.2605.30611.

Dawei Zhu, Rui Meng, Yale Song, Xiyu Wei, Sujian Li, Tomas Pfister, and Jinsung Yoon. Paperbanana: Automating academic illustration for AI scientists. *CoRR*, abs/2601.23265, 2026. doi: 10.48550/ARXIV.2601.23265. URL https://doi.org/10.48550/arXiv.2601.23265.

A LIMITATIONS

DiffBench has several limitations that leave room for future extensions. First, the current benchmark primarily contains English-language documents. Extending it to multilingual documents would enable a broader evaluation of visual difference detection. In addition, future work could also investigate whether detected differences provide effective feedback for downstream agents to successfully correct reconstruction errors.

B THE DETAILS OF DATA

B.1 REFERENCE DOCUMENT SELECTION

We collect a large pool of real-world documents from publicly available PDF and PowerPoint files and render individual pages or slides into images as candidate references. Our goal is to construct a visually diverse and high-quality reference set that covers a broad range of document types, layout and styles.

Since presentation slides differ substantially from other document pages in their composition and design conventions, we perform selection separately for presentation slides and other documents. Specifically, we design two MLLM-based selection prompts tailored to these two sources. For document pages, the MLLM characterizes each candidate according to properties such as document type, layout structure, color palette, typography, visual assets, information density, reading-order complexity, and reconstruction difficulty. For presentation slides, we additionally consider slide-specific characteristics such as slide function, presentation style, information hierarchy, diagrams, charts, screenshots, shape treatments, and slide composition. The complete prompts are provided in Figures 8 and 9.

Based on these structured visual characteristics, the MLLM assigns each candidate a diversity score from 1 to 5 together with a recommendation on whether it should be retained. We shortlist all candidates with a diversity score of 4 or above and subsequently conduct manual review to remove low-quality, redundant, or unsuitable samples while maintaining diversity across document types, layouts, and visual styles. This process results in 578 high-quality and visually diverse reference images, which form the reference set of DiffBench.

B.2 CANDIDATE IMAGE CONSTRUCTION AND SELECTION

To construct realistic candidate images, we use five frontier MLLMs to independently reconstruct each selected reference image: Gemini-3.1-Pro-Preview (Google DeepMind, 2026a), GPT-5.5 (OpenAI, 2026a), GPT-5.6 Sol (OpenAI, 2026b), Grok-4.3 (SpaceXAI, 2026a), and Qwen3.5-122B-A10B (Qwen Team, 2026a). Rather than manually injecting visual perturbations, we generate candidate images directly through vision-to-code reconstruction, allowing the resulting differences to reflect errors that naturally arise from current multi-modal models.

For each reference, the model is provided with the page screenshot, its original dimensions, and the available visual assets extracted from the source PDF for better reconstruction. We instruct the model to reproduce the reference as a single self-contained HTML file, with explicit constraints on page geometry, layout construction, typography, asset placement, and visual styling. The generated HTML is then rendered under a consistent browser environment to obtain the candidate image. The complete reconstruction prompt is shown as Figure 10.

Since different models may produce highly similar reconstructions for the same reference, retaining all outputs would introduce substantial redundancy into the benchmark. We therefore compare candidate images associated with the same reference and remove visually redundant reconstructions while preserving outputs that exhibit distinct reconstruction errors. After this filtering process, we retain 578 candidate images for constructing the final reference-candidate pairs.

B.3 Element Reference Discovery

To provide a consistent set of visual anchors for subsequent difference discovery, we construct an element-level reference set for every reference image. The procedure consists of three stages: independent element discovery, semantic merging, and element localization.

Independent Element Discovery. We use three MLLMs, GPT-5.6 Sol (OpenAI, 2026b), Gemini-3.8-Flash (Google DeepMind, 2026b), and Grok-4.6 (SpaceXAI, 2026b), to independently inventory visible elements in each reference image. Each model receives the complete page image and is instructed to identify concrete and independently referable elements, including text blocks, headings, images, illustrations, charts, icons, logos, shapes, and standalone structural lines. We explicitly control annotation granularity to avoid both excessive fragmentation and overly coarse grouping. For example, contiguous paragraphs are treated as single elements, while unified embedded images are not further decomposed into their internal contents. Abstract regions, whitespace, and invisible grouping containers are excluded.

We additionally instruct each model to perform a dedicated inspection for structural lines, including dividers, separators, rules, underlines, and simple connectors, since such subtle elements can be important for fine-grained visual comparison but are easily omitted during generic visual inspection. At this stage, models output only concise textual descriptions of elements, without coordinates, categories, or confidence scores. The three generation prompts are shown in Figure 11.

Semantic Merging. The independently generated inventories may contain duplicate descriptions, different naming conventions, or elements detected by only a subset of the models. We therefore anonymize the three inventories and assign temporary member IDs before providing them to Gemini-3.8-Flash for semantic merging. The merger clusters descriptions only when they refer to the same physical visible element and produces a single canonical description for each cluster. Nearby but distinct elements, repeated instances, parent-child pairs, and elements internal to a unified image are explicitly prevented from being merged. The merger is also prohibited from introducing any element that does not appear in the source inventories. This produces a canonical element set with consistent semantic references while preserving distinct visible instances.

Element Localization. We next use Gemini-3.8-Flash (Google DeepMind, 2026b) to localize every canonical element on the original reference image. The canonical set is frozen during this stage: the localization model may not add, delete, merge, split, or rename elements. For each canonical element, the model predicts either `located`, `not_found`, or `ambiguous`. Successfully localized elements are assigned a bounding box in normalized image coordinates $[y_{\min}, x_{\min}, y_{\max}, x_{\max}]$ on a 0-1000 scale. These localized element references provide spatial anchors for subsequent difference discovery and facilitate manual verification of element identity and localization quality. The merging and localization prompts are provided in Figures 12 and 13.

B.4 Agentic Hierarchical Difference Discovery

We instantiate three independent agents using GPT-5.6 Sol (OpenAI, 2026b), Gemini-3.8-Flash (Google DeepMind, 2026b), and Grok-4.6 (SpaceXAI, 2026b) in agentic model with high reasoning efforts. Each agent follows the same region-guided visual difference discovery skill and independently analyzes every reference-candidate pair.

The skill first partitions the reference image into coherent visual regions based on the localized elements, layout structure, alignment, spacing, and reading order, and then maps each region to its corresponding area in the candidate image. For each region, the agent constructs complementary spatial and mapped comparison views to inspect both positional changes and within-region content or appearance differences. It then searches for discrepancies at region, cross-region, and whole-page levels, while avoiding redundant descriptions of the same underlying visual event.

Finally, each agent performs a fresh-context verification pass over its proposed differences to remove unsupported or duplicated claims, correct inaccurate descriptions, recover missed differences, and consolidate cascaded effects. The complete skill specification is shown as Figure 14.

B.5 Multi-Model Verification and Consolidation

Multi-Model Verification. Each discovered difference candidate is independently verified by three MLLMs: GPT-5.6 Sol (OpenAI, 2026b), Gemini-3.8-Flash (Google DeepMind, 2026b), and Grok-4.6 (SpaceXAI, 2026b), all with high reasoning effort. Given the corresponding visual evidence and relevant element references, each verifier determines whether the proposed difference is fully supported by the reference-candidate pair. We retain a candidate if it receives a `keep` decision from more than half of the verifiers. The verifier is instructed to judge only the proposed difference and not to discover, correct, merge, or reinterpret additional discrepancies.

Final Consolidation. After majority voting, the retained differences from all discovery agents are merged using GPT-5.6 Sol (OpenAI, 2026b). This stage is text-only and does not re-evaluate the visual evidence. The model clusters only semantically equivalent descriptions that refer to the same subject and visual change, while preserving differences that describe distinct properties or downstream effects. For each cluster, the most precise existing description is selected as the representative without rewriting its content. The resulting groups form the final golden difference set.

B.6 Difference Dimension Annotation

To support dimension-wise evaluation, we annotate each golden difference with one or more of the four evaluation dimensions: *Layout*, *Relation*, *Content Integrity*, and *Visual Properties*. Since a single visual discrepancy may explicitly involve multiple aspects, we adopt a multi-label annotation scheme and require the annotator to provide a short rationale for every assigned dimension. The definition of each dimension and corresponding sub-dimensions is shown as Table 2.

To avoid excessive tagging, the annotator follows an explicit-evidence principle: a dimension is assigned only when the corresponding aspect is directly described in the golden difference, rather than being a potential downstream consequence. We further distinguish intrinsic visual changes from semantic content loss. For example, changes in size, typography, or framing alone are treated as visual properties, while *Content Integrity* is assigned when semantic content is explicitly missing, clipped, truncated, obscured, or otherwise inaccessible. The complete annotation prompts are shown in Figure 15 and Figure 16.

C Additional Analysis of Evaluation Results

C.1 Additional Subdimension Analysis

Content Integrity and Visual Properties. As shown in Figure 6, *Text & Data* is generally the strongest Content Integrity sub-dimension across models, suggesting that explicit textual and numerical discrepancies are easier to identify than changes in visual or structured content. In contrast, *Visual Content* is relatively challenging for most models, indicating greater difficulty in comparing semantic changes in images and illustrations.

A substantially different pattern emerges for *Visual Properties*. Except for GPT-6 Astra OpenAI (2026c), all models perform poorly across nearly every sub-dimension, with Strict Accuracy mostly remaining below 16%. This broad degradation suggests that the difficulty is not confined to a particular property type, but reflects a more general weakness in fine-grained appearance comparison. Among these properties, *Size & Shape* is often relatively easier, whereas typography remains particularly difficult for several models.

C.2 Sensitivity to Subtle Visual Differences

To examine whether models can attend to fine-grained visual details, we measure *Minor Element Coverage*. During the element reference discovery stage mentioned in Section 3.2, each visual element is assigned a unique ID and localized with a bounding box, allowing each golden difference to be traced back to the element(s) it affects. We first regard an element as spatially minor when its bounding box occupies less than 20% of the page, corresponding to an IoU below 0.2 with the full-page region. We select this threshold based on manual inspection, where it provides good agreement with human judgments of visually localized minor elements. And then we collect the unique minor

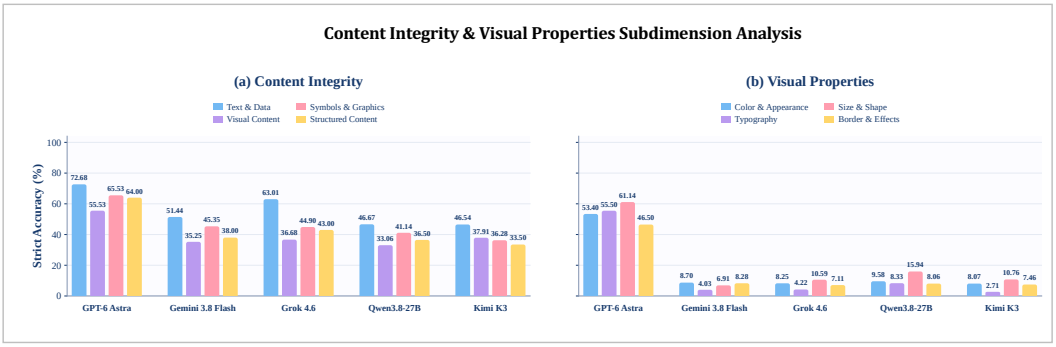

Figure 6: **Sub-dimension analysis of Content Integrity and Visual Properties.** Left: Strict Accuracy across Text & Data, Visual Content, Symbols & Graphics, and Structured Content. Right: Strict Accuracy across Color & Appearance, Typography, Size & Shape, and Border & Effects.

elements mentioned in the golden differences and correct prediction from each model, and regard an element as covered if at least one matched model prediction detects a difference involving that element. Each element is counted only once even if it participates in multiple golden differences, and the coverage is computed as the proportion of covered elements among all the minor elements involved in golden differences.

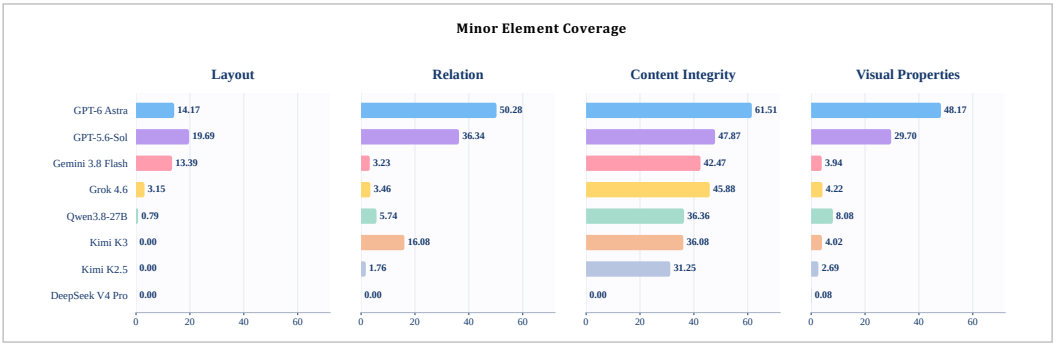

Figure 7: **Minor Element Coverage.** Coverage of unique elements involved in Minor Elements Mentioned in Golden Difference across four evaluation dimensions for representative models.

As shown in Figure 7, *Content Integrity* generally achieves the highest coverage, whereas *Layout* remains below 20% for all models. Outside the GPT family (OpenAI, 2026c;b), coverage reaches at most 16.08% for *Relation* and 8.08% for *Visual Properties*. Even GPT-6 Astra (OpenAI, 2026c) covers only 50.28% and 48.17% of the relevant elements in these two dimensions, respectively. These results indicate that current MLLMs still have limited coverage of fine-grained visual discrepancies. Many small visual targets remain completely undetected, particularly for relational and appearance-related changes, even for the strongest models.

System Prompt: MLLM-based PDF Page Selection

 # Role
You are a visual design and layout annotator for PDF page selection.

 # Task
Analyze the given PDF page screenshot. The goal is to select a visually diverse set of pages for evaluating or training multimodal models on visually rich document understanding, layout perception, and document reconstruction.

 # Diversity Criteria
The final dataset should be diverse across:

- document type

- layout structure
- color palette
- typography
- visual assets
- background / overlay effects
- information density
- structured elements such as tables, charts, maps, forms, and diagrams
- spatial and rendering difficulty

 # Focus
Focus on:

- visual style
- layout structure
- color palette
- typography
- image, table, chart, map, and form elements
- background and overlay effects
- spatial/layout difficulty
- reconstruction difficulty for PDF-to-image or PDF-to-HTML tasks

Do NOT focus on summarizing the document content.
Do NOT transcribe all text.
Do NOT judge whether the document content is interesting.
Only describe what is visually important for diversity, layout understanding, and reconstruction difficulty.
Use multi-labels when appropriate.
If a category is not present, use `"none"`.
If uncertain, use `"uncertain"`.
Return valid JSON only.

 # High-Diversity Indicators
A page should receive a high diversity score if it contains rare or visually challenging features such as:

- complex multi-column layout
- overlapping elements
- semi-transparent overlays
- text wrapping around images
- dense tables
- charts or numeric-heavy layouts
- maps or route diagrams
- forms or floor plans
- unusual typography
- decorative backgrounds or watermarks
- full-bleed photography
- strong color/style identity
- complex image-text interaction
- small captions or dense labels
- irregular, poster-like, or collage-like composition

 # Allowed Labels
document_type:
```
["academic_paper", "research_report", "institutional_report",
"newsletter", "newspaper", "magazine_spread", "brochure",
"flyer", "poster", "cover", "book_cover", "product_catalog",
"technical_specification_sheet", "real_estate_ad", "advertisement",
"menu", "form", "letter", "ppt_slide", "map_page", "calendar",
"university_document", "corporate_sustainability_campaign",
"cartoon_illustration_cover", "other"]
```
layout_structure:
```
["single_column", "two_column", "three_column", "multi_column",
"asymmetric_columns", "strict_grid", "loose_grid",
"modular_blocks", "freeform_collage", "editorial_collage",
"poster_composition", "left_right_split", "top_bottom_split",
"centered_composition", "diagonal_composition", "full_bleed_image",
```

"large_negative_space", "text_wrapping_around_images",
"image_background", "overlapping_elements", "layered_elements",
"tilted_cards", "paper_card_layout", "table_layout", "form_layout",
"map_centered_layout", "calendar_layout", "product_option_grid",
"specification_sheet_grid", "uncertain", "other"]

color_palette:
["grayscale", "monochrome", "duotone", "limited_palette", "colorful",
"highly_colorful", "pastel", "muted", "vivid", "neon", "earth_tone",
"warm_earth_tone", "rustic_brown", "natural_green", "corporate_blue",
"institutional_grey", "luxury_muted", "festive_bright",
"warm_sunset", "dark_background", "light_background", "high_contrast",
"low_contrast", "off_white", "mixed", "uncertain", "other"]

typography:
["serif", "sans_serif", "mixed_serif_sans", "script_font",
"handwritten_font", "comic_lettering", "condensed_font",
"display_font", "monospace", "large_title", "small_refined_type",
"letter_spaced_uppercase", "bold_poster_type", "italic_text",
"rotated_text", "vertical_text", "dense_body_text", "caption_text",
"tiny_labels", "strong_hierarchy", "medium_hierarchy",
"weak_hierarchy", "centered_text", "justified_text",
"mixed_alignment", "uncertain", "other"]

visual_assets:
["none", "single_photo", "multiple_photos", "photo_collage",
"full_bleed_photo", "illustration", "cartoon_illustration",
"icons", "logo", "badge", "clip_art", "cut_out_figure", "map",
"route_lines", "chart", "bar_chart", "line_chart", "pie_chart",
"infographic", "research_figure", "diagram", "table", "dense_table",
"floor_plan", "calendar", "form_fields", "checkboxes", "qr_code",
"decorative_border", "watermark", "product_photography",
"product_thumbnails", "color_swatches", "technical_drawings",
"uncertain", "other"]

background_effects:
["plain_white", "off_white", "solid_color", "colored_blocks",
"gradient", "photo_background", "full_bleed_photo_background",
"patterned_background", "textured_background",
"semi_transparent_overlay", "vertical_translucent_panel",
"tinted_overlay", "dark_overlay", "light_overlay", "watermark",
"drop_shadow", "paper_card_effect", "tilted_card_effect",
"decorative_background", "none", "uncertain", "other"]

information_density:
["very_low", "low", "medium", "high", "very_high"]

visual_density:
["very_low", "low", "medium", "high", "very_high"]

reading_order_difficulty:
["linear", "moderate", "complex", "ambiguous"]

reconstruction_difficulty:
["low", "medium", "high", "very_high"]

aesthetic_style:
["minimalist", "corporate", "academic", "editorial", "luxury",
"rustic", "playful", "festive", "vintage", "technical",
"institutional", "newspaper_like", "scrapbook", "catalog_like",
"template_based", "desktop_publishing", "poster_like",
"magazine_style", "brochure_style", "luxury_travel_editorial",
"community_newsletter", "formal_institutional_flyer",
"real_estate_brochure", "technical_product_catalog",
"corporate_sustainability", "cartoon_comic_style", "eco_campaign",
"other"]

special_layout_features:
["none", "semi_transparent_color_overlay", "text_on_photo",
"white_text_on_dark_overlay", "full_bleed_photo",
"large_negative_space", "overlapping_photos", "cut_out_character",

"tilted_infographic_card", "dense_modular_grid",
"product_option_panels", "map_with_routes", "text_wrapping",
"watermark", "decorative_clipart", "poster_title",
"floor_plan_layout", "table_heavy", "form_heavy", "chart_heavy",
"small_multiples", "complex_captioning", "irregular_image_crops",
"uncertain", "other"]

 # Output Format

Now analyze the given page screenshot and output the following JSON schema only:

```
{
  "page_id": "<PDF_ID_PAGE_NUMBER>",
  "short_visual_description": "",
  "document_type": [],
  "layout_structure": [],
  "color_palette": [],
  "typography": [],
  "visual_assets": [],
  "background_effects": [],
  "aesthetic_style": [],
  "special_layout_features": [],
  "dominant_colors": [],
  "notable_design_features": [],
  "information_density": "",
  "visual_density": "",
  "reading_order_difficulty": "",
  "reconstruction_difficulty": "",
  "difficulty_reasons": [],
  "diversity_value": {
    "score_1_to_5": 0,
    "reason": ""
  },
  "recommended_use": {
    "keep_for_diversity": true,
    "reason": ""
  },
  "possible_failure_modes_for_mllm_or_reconstruction": []
}
```

Figure 8: The system prompt used for MLLM-based PDF page characterization and selection. The model identifies visual and structural attributes, estimates reconstruction difficulty, and assigns a diversity score to support the selection of visually diverse document pages.

System Prompt: MLLM-based PowerPoint Slide Characterization and Selection

 # Role

You are a visual design and layout annotator for PowerPoint slide selection.

 # Task

Analyze the given slide screenshot. The goal is to select a visually diverse set of slides for evaluating or training multimodal models on slide understanding, visual layout perception, and slide reconstruction.

 # Diversity Criteria

The final dataset should be diverse across:

- slide function
- presentation style
- layout structure
- visual hierarchy
- color theme
- typography
- diagrams, charts, tables, screenshots, icons, and images
- background and shape treatment
- information density

- spatial and reconstruction difficulty

 # Focus
Focus on:

- visual style
- slide composition
- information hierarchy
- layout structure
- color palette
- typography
- visual assets
- chart, table, diagram, and screenshot elements
- background, shapes, overlays, and shadows
- reconstruction difficulty for PPT-to-image or slide-to-code tasks

Do NOT summarize the slide content.
Do NOT transcribe all text.
Do NOT judge whether the content is correct.
Only describe what is visually important for diversity, layout understanding, and reconstruction difficulty.
Use multi-labels when appropriate.
If a category is not present, use `"none"`.
If uncertain, use `"uncertain"`.
Return valid JSON only.

 # High-Diversity Indicators
A slide should receive a high diversity score if it contains rare or visually challenging features such as:

- complex diagram or flowchart
- timeline, roadmap, matrix, pyramid, funnel, or framework
- dashboard-like layout
- dense charts or tables
- annotated screenshots or UI mockups
- many icons or visual cards
- overlapping elements
- semi-transparent overlays
- gradient or photo background
- strong branding or unusual color theme
- very small labels or captions
- strong typography hierarchy
- large negative space
- complex alignment or spacing
- non-standard slide composition

 # Allowed Labels
slide_type:
`["title_slide", "section_divider", "agenda_slide", "overview_slide", "content_slide", "bullet_slide", "comparison_slide", "process_slide", "timeline_slide", "roadmap_slide", "framework_slide", "matrix_slide", "chart_slide", "table_slide", "dashboard_slide", "case_study_slide", "quote_slide", "image_hero_slide", "product_intro_slide", "product_demo_slide", "team_slide", "academic_slide", "technical_slide", "closing_slide", "thank_you_slide", "other"]`
presentation_style:
`["corporate", "consulting_deck", "academic_presentation", "technical_presentation", "startup_pitch", "product_marketing", "sales_deck", "training_material", "educational", "medical_scientific", "financial_report", "business_strategy", "design_portfolio", "conference_talk", "workshop_slide", "playful", "luxury", "minimalist", "template_based", "other"]`
layout_structure:
`["single_column", "two_column", "three_column", "multi_column", "title_body", "title_plus_visual", "hero_image_with_text", "full_bleed_image", "split_screen", "sidebar_layout", "card_grid",`

"modular_blocks", "quadrant_layout", "centered_composition",
"asymmetric_layout", "diagonal_layout", "layered_layout",
"floating_cards", "timeline_layout", "process_flow_layout",
"cycle_diagram_layout", "matrix_layout", "pyramid_layout",
"funnel_layout", "dashboard_layout", "screenshot_annotation_layout",
"comparison_layout", "before_after_layout", "map_layout",
"large_negative_space", "other"]

information_structure:
["topic_title", "assertion_title", "question_title",
"numbered_points", "bullet_list", "key_message_plus_supporting_details",
"problem_solution", "before_after", "cause_effect", "pros_cons",
"comparison_table", "step_by_step", "hierarchical_structure",
"framework_structure", "storytelling_sequence",
"data_driven_argument", "visual_explanation", "other"]

color_theme:
["light_theme", "dark_theme", "white_background", "black_background",
"corporate_blue", "consulting_blue_grey", "academic_blue",
"tech_dark", "startup_gradient", "pastel", "muted", "vivid",
"high_contrast", "low_contrast", "monochrome", "duotone", "colorful",
"earth_tone", "luxury_muted", "playful_bright", "medical_green_blue",
"financial_blue_grey", "red_warning_accent", "gradient_background",
"brand_color_heavy", "other"]

typography:
["large_title", "small_title", "assertive_headline", "subtitle",
"section_label", "bullet_text", "dense_body_text", "caption_text",
"tiny_labels", "numbered_labels", "bold_keywords", "highlighted_text",
"all_caps", "letter_spaced_uppercase", "serif", "sans_serif",
"mixed_fonts", "display_font", "monospace", "weak_hierarchy",
"medium_hierarchy", "strong_hierarchy", "very_strong_hierarchy",
"other"]

visual_assets:
["none", "single_photo", "multiple_photos", "full_bleed_photo",
"illustration", "icons", "logo", "avatar", "product_image",
"screenshot", "annotated_screenshot", "ui_mockup", "diagram",
"flowchart", "timeline", "roadmap", "cycle_diagram", "pyramid",
"funnel", "matrix", "framework_diagram", "chart", "bar_chart",
"line_chart", "pie_chart", "scatter_plot", "area_chart", "table",
"dashboard_widgets", "map", "research_figure", "equation",
"code_snippet", "callout_box", "arrow", "connector_lines", "badges",
"decorative_shapes", "other"]

background_and_shapes:
["plain_background", "solid_color_background", "gradient_background",
"photo_background", "patterned_background", "geometric_background",
"abstract_shapes", "colored_blocks", "semi_transparent_overlay",
"glassmorphism", "drop_shadow", "card_shadow", "rounded_rectangles",
"thin_divider_lines", "decorative_lines", "large_shape_blocks",
"blob_shapes", "angled_shapes", "layered_shapes", "none", "other"]

text_density:
["very_low", "low", "medium", "high", "very_high"]

visual_density:
["very_low", "low", "medium", "high", "very_high"]

element_count:
["1-3", "4-8", "9-15", "16-30", "30+"]

reading_order_difficulty:
["linear", "moderate", "complex", "ambiguous"]

reconstruction_difficulty:
["low", "medium", "high", "very_high"]

reconstruction_challenges:
["none", "alignment_difficulty", "spacing_sensitivity",
"negative_space_importance", "small_text_difficulty",
"chart_precision_difficulty", "table_structure_difficulty",

"diagram_structure_difficulty", "connector_line_difficulty",
"shape_layering_difficulty", "image_crop_difficulty",
"screenshot_detail_difficulty", "color_matching_difficulty",
"font_matching_difficulty", "brand_template_difficulty",
"background_reconstruction_difficulty", "other"]

 # Output Format
Now analyze the given slide screenshot and output the following JSON schema only:

```
{
  "slide_id": "<PPT_ID_SLIDE_NUMBER>",
  "short_visual_description": "",
  "slide_type": [],
  "presentation_style": [],
  "layout_structure": [],
  "information_structure": [],
  "color_theme": [],
  "dominant_colors": [],
  "typography": [],
  "visual_assets": [],
  "background_and_shapes": [],
  "text_density": "",
  "visual_density": "",
  "element_count": "",
  "reading_order_difficulty": "",
  "reconstruction_difficulty": "",
  "reconstruction_challenges": [],
  "difficulty_reasons": [],
  "notable_design_features": [],
  "diversity_value": {
    "score_1_to_5": 0,
    "reason": ""
  },
  "recommended_use": {
    "keep_for_diversity": true,
    "reason": ""
  },
  "possible_failure_modes_for_mllm_or_reconstruction": []
}
```

Figure 9: The system prompt used for MLLM-based PowerPoint slide characterization and selection. The model identifies visual, structural, and stylistic attributes, estimates reconstruction difficulty, and assigns a diversity score to support the selection of visually diverse and challenging slides.

Prompt: Vision-to-Code Candidate Reconstruction

 # System Prompt
You are an expert HTML/CSS developer. Think step by step about the layout structure before writing code.

 # Reconstruction Prompt
Reproduce the page screenshot as a single self-contained HTML file. You have the following image assets extracted from the original PDF. Mark their positions with placeholder divs.

Available assets:
{asset_list}

 # Critical Dimensions

- Page width: {page_width}px
- Page height: {page_height}px
- Aspect ratio: {aspect_ratio}

 # Rules

1. `body {{ margin: 0; padding: 0; }}`. The root container must be exactly {page_width}px × {page_height}px.
2. Use CSS flexbox or CSS grid for ALL layout positioning.
3. Use a single `<style>` block with semantic classes. Do not use Tailwind or inline styles.
4. Do NOT use `<br>` for text wrapping.
5. For each image, chart, or figure position, use `<div class="asset-placeholder" data-asset="img_XX.png"></div>` with the correct asset filename. Give it approximate width and height in CSS to reserve the space.
6. If no asset matches a visual element, use a plain colored `<div>` placeholder without `data-asset`.
7. Use system fonts only (Arial, Georgia, Times New Roman). Do not use `@import`.
8. Do not include HTML/CSS comments.
9. Match font sizes, weights, colors, and spacing precisely.
10. Reproduce decorative elements such as borders, shadows, and gradients with CSS.
11. Not all assets may be needed. Only reference an asset if you can clearly identify where it belongs.

 # Output
Output only the complete HTML file.

Figure 10: The prompt used to generate candidate reconstructions from selected reference images. Each MLLM receives the reference screenshot, page geometry, and available visual assets, and produces a self-contained HTML reconstruction.

Prompts: Independent Element Inventory

 # GPT-5.6-Sol
You are independently inventorying concrete visual elements in one complete page image.
Scan from top to bottom and left to right. Return every independently referable visible element that would help another vision model discuss this page: exact readable text blocks, headings, labels, icons, logos, photographs, unified embedded images, diagrams, controls, colored shapes, and standalone structural lines.
Use balanced granularity: - Keep each contiguous body paragraph or text block together; do not split words or text lines. - Never output the complete page/canvas, whitespace, an abstract region, column, or invisible grouping container. - Treat a clearly bounded embedded raster image, photograph, collage, screenshot, poster reproduction, chart, or illustration as one element. Do not enumerate content inside a unified embedded image. - Do not output both a parent enclosure and its children. Omit a rectangle or border whose only role is to contain separately listed content. - Make a dedicated pass for lines. A line is a simple, standalone layout stroke whose main visual identity is the line itself. Include faint, thin, short, thick, horizontal, vertical, diagonal, curved, solid, dashed, or dotted dividers, separators, rules, underlines, signature or fill-in lines, and simple connector lines. A simple rectangular outline or frame is one line element, not four separate edges. Exclude table or cell gridlines, chart axes or gridlines, text strokes, image edges, and strokes that belong to a map, route, icon, logo, illustration, diagram, background graphic, decorative pattern, or other graphic. Filled bars or bands, small squares, geometric marks, and decorative shapes are not lines. - Distinguish repeated elements by visible position or order. - Preserve readable text exactly; use [unclear] instead of guessing. - Describe only visible facts, not hidden purpose or functionality.
Each description must be concise, self-contained, and uniquely locatable from the full image. Do not output IDs, coordinates, bounding boxes, confidence, categories, or reasoning.
Return exactly one JSON object and no surrounding prose: ”elements”:[”description”, ”description”]

 # Gemini-3.8-Flash
Create an independent inventory of meaningful visible elements in the supplied complete page image.
Use this inspection order: 1. Read the page top-to-bottom and capture exact headings, labels, captions, and contiguous body-text blocks. 2. Capture logos, icons, controls, photographs, unified embedded raster images, diagrams, charts, and independently meaningful shapes. 3. Make a separate pass for lines. A line is a simple, independently visible layout stroke whose primary form is the line itself. Capture faint, thin, short, thick, horizontal, vertical, diagonal, curved, solid, dashed, or dotted dividers, separators, rules, underlines, signature or fill-in lines, and simple connector lines. Treat a simple rectangular outline or frame as one line element rather than four edges. Do not treat table or cell gridlines, chart axes or gridlines, text strokes, image edges, or strokes within a map, route, icon, logo, illustration, diagram,

background graphic, decorative pattern, or other graphic as lines. Filled bars or bands, small squares, geometric marks, and decorative shapes are not lines.
Granularity rules: - A contiguous paragraph is one element, not one element per line or word. - Never describe the whole input canvas, whitespace, columns, sections, abstract regions, or invisible containers as elements. - A clearly bounded photograph, collage, screenshot, poster reproduction, chart, table, or illustration is one element. Do not enumerate content inside a unified embedded image. - Choose one semantic level: never output both a parent enclosure and child elements for the same content. Omit decorative enclosing rectangles when their inner elements are listed. - Do not add any other kind of stroke as a line element; shadows and compression artifacts are also excluded. - Preserve readable text exactly and use [unclear] when necessary. - Distinguish repeated elements by location or order.
Write short, visually grounded descriptions with enough visible position information to uniquely identify each element. Output descriptions only—no IDs, coordinates, bounding boxes, categories, confidence, or explanation.
Return exactly: "elements":["description", "description"]

 # Grok-4.6
Independently inspect this complete page image and inventory concrete, visible elements that another vision model could refer to when describing fine-grained visual differences.
Scan from top to bottom and left to right. Include exact readable text blocks, headings, labels, icons, logos, photographs, unified embedded images, diagrams, controls, meaningful shapes, and independently visible structural lines.
Use restrained element-level granularity: - Keep each contiguous paragraph or body-text block together; never split it into words or individual text lines. - Never output the complete page/canvas, whitespace, abstract regions, columns, sections, or inferred grouping containers. - Treat each clearly bounded photograph, collage, screenshot, poster reproduction, table, chart, or illustration as one element. Do not enumerate objects or text inside a unified embedded image. - Use one semantic level only. Do not output a parent enclosure when its meaningful children are already listed. - Inspect lines separately. A line is a simple, standalone layout stroke whose visual identity is the line itself. Include faint, thin, short, thick, horizontal, vertical, diagonal, curved, solid, dashed, or dotted dividers, separators, rules, underlines, signature or fill-in lines, and simple connector lines. A simple rectangular outline or frame is one line element, not four separate edges. Exclude table or cell gridlines, chart axes or gridlines, text strokes, image edges, and strokes belonging to a map, route, icon, logo, illustration, diagram, background graphic, decorative pattern, or other graphic. Filled bars or bands, small squares, geometric marks, decorative shapes, shadows, and compression artifacts are not lines. - Preserve readable text exactly; use [unclear] rather than guessing. - Distinguish repeated elements by visible location or order. - Describe only visible facts, never inferred intent or functionality.
Each description must be concise, self-contained, and uniquely locatable on the full page. Return descriptions only: no IDs, coordinates, bounding boxes, confidence, categories, or reasoning.
Return exactly one JSON object with no surrounding prose: "elements":["description", "description"]

Figure 11: Prompts used for independent element discovery with GPT-5.6-Sol, Gemini-3.8-Flash, and Grok-4.6. Each model independently inventories concrete and referable visual elements from the complete reference image without producing spatial coordinates.

Prompt: Semantic Merging of Element Inventories

 # Role and Task
You are merging three anonymous, independently produced visual-element inventories for one complete page image.
Your task is semantic equivalence clustering and canonical naming only. Do not generate bounding boxes. Do not add any element that is absent from all source lists.

 # Clustering Rules

- Put source mentions in the same cluster only when they refer to the same physical visible element.
- Do not merge nearby but separate elements, repeated instances, a parent with its child, a complete image with an internal item, or a chart/table with one of its labels.
- Every input member ID must appear in exactly one output cluster.
- A cluster may contain several mentions from one source, but this does not imply cross-source support.
- Keep singleton clusters; deterministic code will apply the consistency threshold later.

 # Canonical Naming Rules

- Create one concise, self-contained canonical name for the physical element.
- Preserve exact readable text when source descriptions agree.
- Include visible distinguishing properties and location when needed.
- Never use abstract names such as region, content area, section, or container.
- Do not mention source identities or model behavior.

 # Output Format

```
{
  "clusters": [
    {
      "canonical_id": "c001",
      "canonical_name": "description",
      "member_ids": ["A001", "B004"]
    }
  ]
}
```

Canonical IDs must follow top-to-bottom and then left-to-right visual order.

Figure 12: Prompt used by Gemini-3.8-Flash to semantically merge the three independent element inventories into a canonical element reference set.

Prompt: Canonical Element Localization

 # Task

Locate a frozen list of canonical visual elements in the supplied complete page image.
You may only localize the supplied IDs. Do not add, delete, merge, split, rename, or reinterpret elements.
Return exactly one result for every supplied `canonical_id`.
Coordinates use integer $[y_{\min}, x_{\min}, y_{\max}, x_{\max}]$ normalized to 0–1000 relative to the complete provided image canvas.

 # Localization Rules

- `located`: the element is clearly visible and uniquely locatable.
- `not_found`: the element is absent or cannot be reliably located.
- `ambiguous`: multiple visible instances match the canonical description.
- The bounding box must be `null` unless the status is `located`.
- A localized box must fully enclose the visible element while excluding unrelated neighboring elements.
- Text boxes must include complete glyphs and punctuation.
- Line boxes must include the complete stroke and both endpoints.
- Photographs, cards, charts, and shapes should be aligned to their visible outer boundaries.

 # Output Format

```
{
  "canonical_set_sha256": "<HASH>",
  "elements": [
    {
      "canonical_id": "c001",
      "status": "located",
      "bbox": [0, 0, 100, 100]
    }
  ]
}
```

Figure 13: Prompt used by Gemini-3.8-Flash to localize the frozen canonical element set on the reference image using normalized bounding-box coordinates.

Skill: Region-Guided Visual Difference Discovery

 # Overview
Given a reference page, a candidate page, and localized reference elements, first divide the reference page into coherent regions and map each region to the candidate. Then discover visual differences within regions, across regions, and at the whole-page level. Finally, verify the proposed differences, remove duplicates and redundant cascade descriptions, and assign severity.

 # Inputs
- A native-resolution reference screenshot.
- A native-resolution candidate screenshot.
- Reference elements with stable IDs, visible descriptions, and bounding boxes in the native reference coordinate system.

Element bounding boxes localize elements but do not define region boundaries. Existing difference annotations and evaluation rubrics are not used.

 # Region Division
Divide the page using only the reference screenshot and reference elements. Do not inspect the candidate until the reference division is complete and frozen.
Represent the page as one `page_scope` and a flat set of visually coherent regions.
- Use visible containers, backgrounds, columns, rows, alignment, spacing, dividers, repetition, and reading flow as grouping evidence.
- Choose the smallest coherent module that preserves the context needed for comparison.
- Do not create regions around individual words, text lines, or small objects unless they form standalone modules.
- Keep elements together when they share a container, alignment system, spacing rhythm, or reading flow.
- Split modules when they have independent boundaries, gutters, backgrounds, alignment systems, or repeated-unit structures.
- Assign every reference element to exactly one primary region.
- Avoid overlapping regions unless visual layering makes overlap unavoidable.
- Choose region boundaries from the visible composition rather than mechanically unioning element boxes.

Use `page_scope` only for complete-page properties. If an important relationship spans multiple regions, record it as a `relation_candidate`.

 # Region Mapping
After freezing the reference division, inspect the candidate and map every reference region as a complete unit. Preserve all reference region IDs, roles, element memberships, and reference boxes.
Use visible content, role, container, surrounding modules, and reading-flow position to establish correspondence.
Assign one mapping status:
- `matched`: the complete corresponding module is identified;
- `partial`: only part of the module is identified;
- `unmapped`: no reliable correspondence exists;
- `ambiguous`: several candidate areas are plausible.

 # Visual Evidence
For every region, construct:
- a **spatial comparison**, using the reference box at the same numerical coordinates in both images;
- a **mapped comparison**, comparing the reference region with its mapped candidate counterpart.

Retain complete-page comparisons for page dimensions, aspect ratio, overall composition, margins, clipping, and overflow.

 # Difference Generation
Each difference has one primary scope:
- `region`: changes within one coherent region;
- `cross_region`: relationships involving multiple regions;

- `whole_page`: page geometry and page-wide composition.

Use the narrowest scope that fully describes the visible event. Do not report the same event at multiple scopes.
For region-level inspection, check color, typography, borders, containment, overlap, relative size, text wrapping, alignment, stacking, padding, gutters, and repeated spacing.
For cross-region inspection, check shared edges, row or column alignment, gutters, ordering, overlap, containment, relative sizing, and repeated-region distribution.
For whole-page inspection, compare native page dimensions, aspect ratio, background, framing, composition, margins, whitespace, clipping, and overflow.
When all children share one coherent movement, resize, or reflow, report one region-level difference instead of repeating the event for every child. Distinct visible consequences such as overlap, clipping, or separately broken alignment should be retained.

 # Verification
Verify all proposed differences in a fresh context with no access to the generation conversation or reasoning. Supply only the visual evidence, frozen region information, element references, and proposed differences.
Choose one action for each proposal:

- `keep`: visually supported and correctly stated;
- `remove`: absent, unsupported, trivial, or duplicated;
- `correct`: supported but inaccurately described;
- `add`: visibly present but missing.

When multiple records describe the same event, keep the most complete one. If a child record only repeats a coherent region-level movement, resize, or reflow, remove it as a cascade. Retain distinct visible consequences and link them to their underlying cause where appropriate.

 # Severity
Assign one severity to every retained difference:

- `major`: substantially affects the overall composition, dominant content, or a large, repeated, or cascading portion of the page;
- `medium`: clearly visible at normal scale and materially affects one region;
- `minor`: localized, low-prominence, weakly visible, non-cascading, and limited in overall impact.

Judge severity from visual impact, prominence, affected area, repetition, and causal reach rather than difference category alone.

Figure 14: The region-guided visual difference discovery skill used by each annotation agent. The skill constructs and maps coherent regions, performs complementary spatial and mapped comparisons, discovers differences across region, cross-region, and whole-page scopes, and verifies the resulting differences in a fresh context.

Prompt: Visual Difference Dimension Annotation

 # Role
You are an expert annotator for a visual difference benchmark.
You are given a golden difference description describing a discrepancy between a reference image and a candidate image.

 # Task

1. Identify all applicable high-level dimensions from the taxonomy below.
2. A single golden difference may receive one or multiple dimension tags.
3. For each assigned dimension, provide a short reasoning explaining why the original golden difference belongs to that dimension.

 # Taxonomy

1. Layout
Differences in the global or page-level structure and composition.

- **Page Geometry**: page dimensions, aspect ratio, or orientation.
- **Layout Skeleton**: major structural regions such as columns, sections, headers, or sidebars.
- **Global Composition**: page-wide positioning, distribution, balance, or alignment of major regions.

Use `layout` when the discrepancy concerns macro structure or page-level composition rather than only an individual element or local spatial relationship.

2. Relation
Differences in spatial relationships between elements or between an element and its surrounding boundary.

- **Position**: absolute or relative positions.
- **Alignment**: shared edges, centers, axes, or baselines.
- **Spacing**: gaps, margins, padding, gutters, or clearance.
- **Topology**: containment, overlap, touching, occlusion, or inside/outside relationships.

Use `relation` when the discrepancy concerns where elements are located or how they spatially interact with other elements or boundaries.

3. Content Integrity
Differences in what semantic content is present or represented.

- **Text & Data**: wording, spelling, punctuation, numbers, dates, URLs, labels, or values.
- **Visual Content**: image or illustration identity and depicted semantic content.
- **Symbols & Graphics**: icons, logos, glyphs, and symbols.
- **Structured Content**: semantic components of charts, tables, diagrams, lists, or other structured representations.

This dimension also includes semantic content that is missing, added, substituted, duplicated, obscured, or otherwise inaccessible.

4. Visual Properties
Differences in the intrinsic visual appearance or geometry of an element while its semantic identity remains unchanged.

- **Color & Appearance**: color, fill, brightness, contrast, opacity, or gradient.
- **Typography**: font family, size, weight, style, letter spacing, line height, or decoration.
- **Size & Shape**: width, height, aspect ratio, scale, or geometric shape.
- **Border & Effects**: borders, strokes, corner radius, shadows, or outlines.

Use `visual_properties` when the discrepancy concerns how an element itself is visually rendered rather than its semantic identity or spatial relationship.

 # Multi-Tag Annotation
A golden difference may belong to multiple dimensions simultaneously. Assign multiple tags only when the original description explicitly contains meaningful discrepancies from multiple dimensions. Do not assign a dimension only because it could be a possible consequence of the described change.

 # Annotation Principles

1. **Use explicit evidence only.** Assign a dimension only when that aspect is directly supported by the golden difference description.
2. **Allow multiple dimensions when multiple aspects are explicitly described.** The four dimensions are not mutually exclusive.
3. **Classify according to the level and nature of the change.** Intrinsic appearance or geometry corresponds to `visual_properties`; spatial organization to `relation`; semantic information or accessibility to `content_integrity`; and macro page structure to `layout`.
4. **Distinguish visual-form changes from semantic content loss.** Changes in line count, size, arrangement, framing, or shape alone do not imply `content_integrity`. Assign it when semantic content is explicitly missing, clipped, truncated, obscured, cut off, or otherwise inaccessible. If such loss is caused by a spatial boundary or overlap, additionally assign `relation`.
5. **Use the minimum sufficient set of tags.** Every assigned dimension must correspond to a directly described and independently meaningful aspect of the difference.

 # Input
Each item below is an independent golden difference. Annotate every `difference_id` exactly once.

```
{differences}
```

 # Output Format
Return a strictly valid JSON array only. Use only the following dimension names: `layout`, `relation`, `content_integrity`, and `visual_properties`.

```
[
```

```
  {
    "difference_id": "g1",
    "dimension_reasoning": [
      {
        "dimension": "relation",
        "reasoning":
          "The discrepancy changes the spatial
          relationship between elements."
      },
      {
        "dimension": "content_integrity",
        "reasoning":
          "The discrepancy also makes semantic content unavailable."
      }
    ]
  }
]
```

Figure 15: Prompt used for multi-label dimension annotation of golden visual differences. The annotator assigns only explicitly supported dimensions and provides a rationale for each assigned label.

Prompt: Visual Difference Sub-dimension Annotation

 # Role
You are an expert annotator for a visual difference benchmark.
Each golden difference has already been assigned one or more high-level dimensions. For each assigned dimension, you are given the original golden difference, the high-level dimension, and the reasoning explaining why that dimension applies.

 # Task
Assign all applicable sub-dimension tags within each already-assigned high-level dimension.
The high-level dimension assignments are fixed. Do NOT change, remove, add, or reconsider any high-level dimension.

 # Taxonomy
1. Layout
- `page_geometry`: page dimensions, aspect ratio, or orientation.
- `layout_skeleton`: major structural organization such as columns, sections, headers, sidebars, or other major page regions.
- `global_composition`: page-wide positioning, distribution, balance, alignment, or arrangement of major regions.

2. Relation
- `position`: absolute or relative positions of elements.
- `alignment`: shared edges, centers, axes, or baselines.
- `spacing`: gaps, margins, padding, gutters, or clearance.
- `topology`: containment, overlap, touching, occlusion, or boundary relationships.
- `flow`: local wrapping, reflow, stacking, or reorganization caused by surrounding elements or available space.

3. Content Integrity
- `text_and_data`: textual or numerical information.
- `visual_content`: images, photographs, illustrations, or other semantic visual assets.
- `symbols_and_graphics`: icons, logos, glyphs, or symbols.
- `structured_content`: charts, tables, diagrams, lists, or other structured representations.

4. Visual Properties
- `color_and_appearance`: color, fill, brightness, contrast, opacity, or gradient.
- `typography`: font family, size, weight, style, letter spacing, line height, or decoration.
- `size_and_shape`: width, height, aspect ratio, scale, or geometric shape.

- `border_and_effects`: borders, strokes, dividers, corner radius, shadows, outlines, or decorative rules.

 # Annotation Principles

1. **Keep the high-level dimension fixed.** Only select sub-dimensions belonging to the provided high-level dimension.
2. **Use both the original difference and dimension reasoning.** The original difference provides the complete discrepancy, while the reasoning indicates which aspect is relevant to the current dimension.
3. **Multiple sub-dimensions are allowed.** Assign multiple tags only when multiple independently meaningful aspects are explicitly described.
4. **Use the minimum sufficient set.** Do not infer potential causes or downstream consequences.
5. **Distinguish relation sub-dimensions carefully.** Use `position` for location, `alignment` for shared spatial references, `spacing` for distances, `topology` for overlap or containment, and `flow` for local wrapping or reorganization.
6. **Distinguish local relation from page-level layout.** Local spatial organization belongs to `relation`, while macro organization of major page regions belongs to `layout`.
7. **Classify content by the affected semantic element.** Missing, added, duplicated, clipped, or obscured content is categorized according to whether it is text/data, visual content, symbols/graphics, or structured content.
8. **Classify visual properties by the intrinsic property.** Select the sub-dimension corresponding directly to the changed visual property.

 # Input

Each item contains a `difference_id`, the original `difference`, and one or more entries in `dimension_reasoning`.

`{differences}`

 # Output Format

Return a strictly valid JSON array only. Preserve every `difference_id`, high-level dimension, and reasoning exactly, and add a `subdimensions` field to every existing dimension entry.

```
[
  {
    "difference_id": "g1",
    "dimension_reasoning": [
      {
        "dimension": "relation",
        "reasoning":
          "The discrepancy changes both the position and alignment...",
        "subdimensions": [
          "position",
          "alignment"
        ]
      },
      {
        "dimension": "visual_properties",
        "reasoning":
          "The discrepancy also changes the intrinsic size...",
        "subdimensions": [
          "size_and_shape"
        ]
      }
    ]
  }
]
```

Do not modify the provided high-level dimensions or reasoning.
Do not add or remove high-level dimensions.
Do not rewrite the golden difference.
Do not output confidence scores or explanations outside the JSON array.

Figure 16: **Prompt for visual difference sub-dimension annotation.** Given fixed high-level dimension labels and their rationales, the annotator assigns the applicable fine-grained sub-dimensions while preserving the original annotations.

Dimension	Definition	Sub-dimension	Definition
Layout	Differences in the global or page-level structure and composition.	Page Geometry	Differences in page dimensions, aspect ratio, or orientation.
		Layout Skeleton	Differences in the major structural organization of the page, such as columns, sections, headers, sidebars, or other major regions.
		Global Composition	Differences in the page-wide positioning, distribution, balance, alignment, or arrangement of major regions.
Relation	Differences in spatial relationships or local organization among elements, or between elements and their surrounding boundaries.	Position	Differences in the absolute or relative positions of elements.
		Alignment	Differences in horizontal or vertical alignment, shared edges, centers, axes, or baselines.
		Spacing	Differences in gaps, margins, padding, gutters, clearance, or other distances between elements or boundaries.
		Topology	Differences in containment, overlap, touching, occlusion, or inside/outside relationships.
		Flow	Differences in how text or other elements locally wrap, reflow, stack, or reorganize in response to surrounding elements or available space.
Content Integrity	Differences in what semantic information is present, represented, or accessible.	Text & Data	Differences in wording, spelling, punctuation, numbers, dates, URLs, labels, or data values.
		Visual Content	Differences in the semantic identity or depicted content of images, photographs, illustrations, or other visual assets.
		Symbols & Graphics	Differences involving icons, logos, glyphs, symbols, or similar semantic graphical elements.
		Structured Content	Differences in the semantic content or components of charts, tables, diagrams, lists, or other structured representations.
Visual Properties	Differences in the intrinsic visual appearance or geometry of an element while its semantic identity remains unchanged.	Color & Appearance	Differences in color, fill, brightness, contrast, opacity, gradient, or similar appearance properties.
		Typography	Differences in font family, font size, weight, style, letter spacing, line height, or text decoration.
		Size & Shape	Differences in an element's width, height, aspect ratio, scale, or geometric shape.
		Border & Effects	Differences in borders, strokes, dividers, corner radius, shadows, outlines, decorative rules, or similar visual effects.

Table 2: **Definition of evaluation dimensions and sub-dimensions.**

D QUALITATIVE EXAMPLES OF DIFFBENCH

In this section, we provide qualitative examples spanning five image categories including presentation slides Figure 17, editorial publications Figure 18, commercial materials Figure 20, scientific posters Figure 19, and educational resources Figure 21. Each example pairs a reference image with a candidate image and gold-standard difference annotations, highlighting discrepancies in content integrity, visual property and relation.

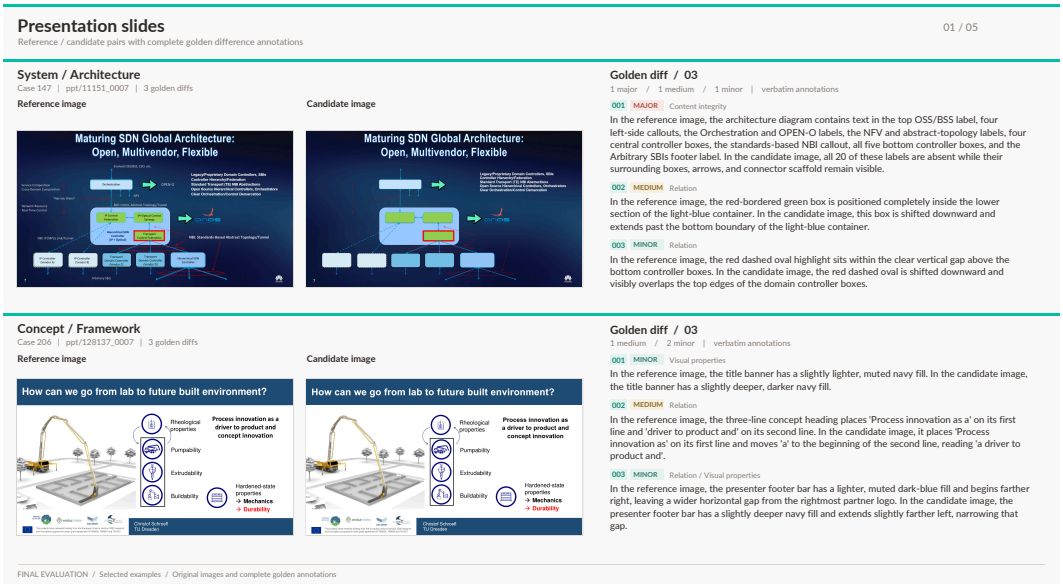

Figure 17: **Qualitative examples of Presentation Slides in DiffBench.**

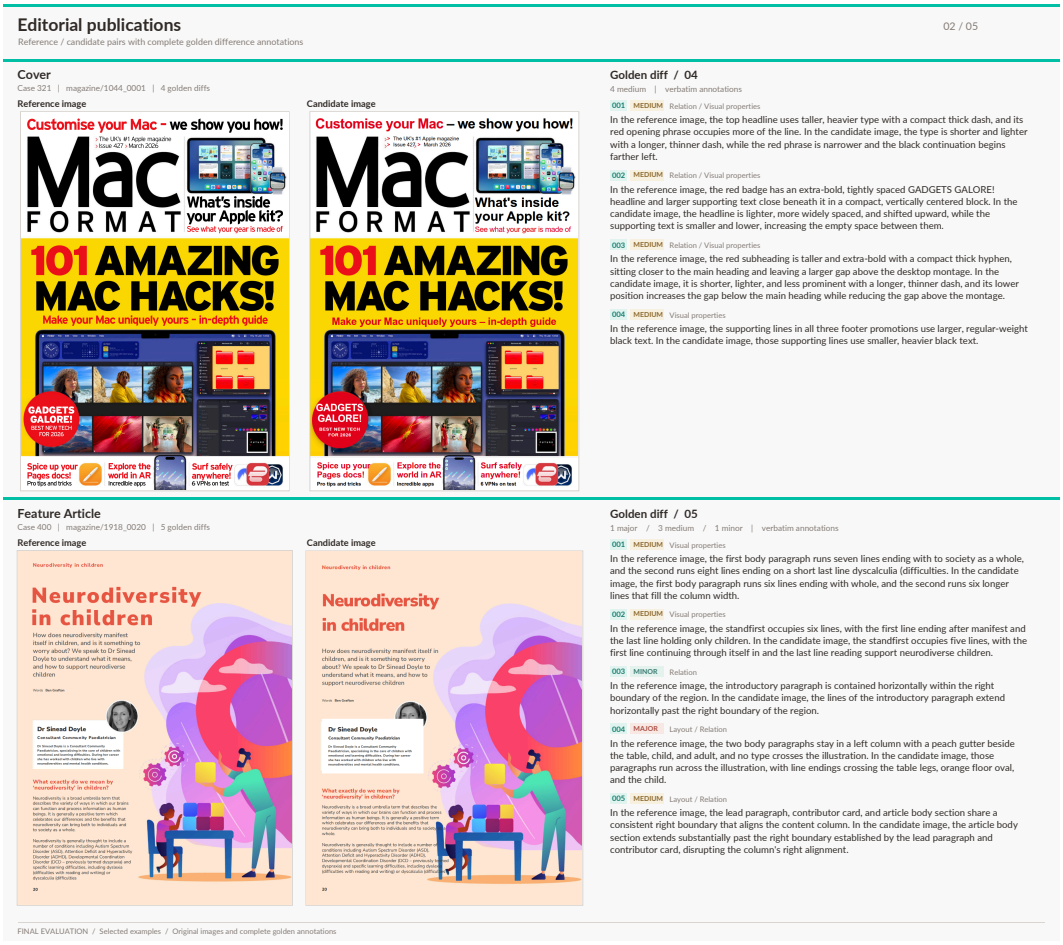

Figure 18: **Qualitative examples of Editorial Publications in DiffBench.**

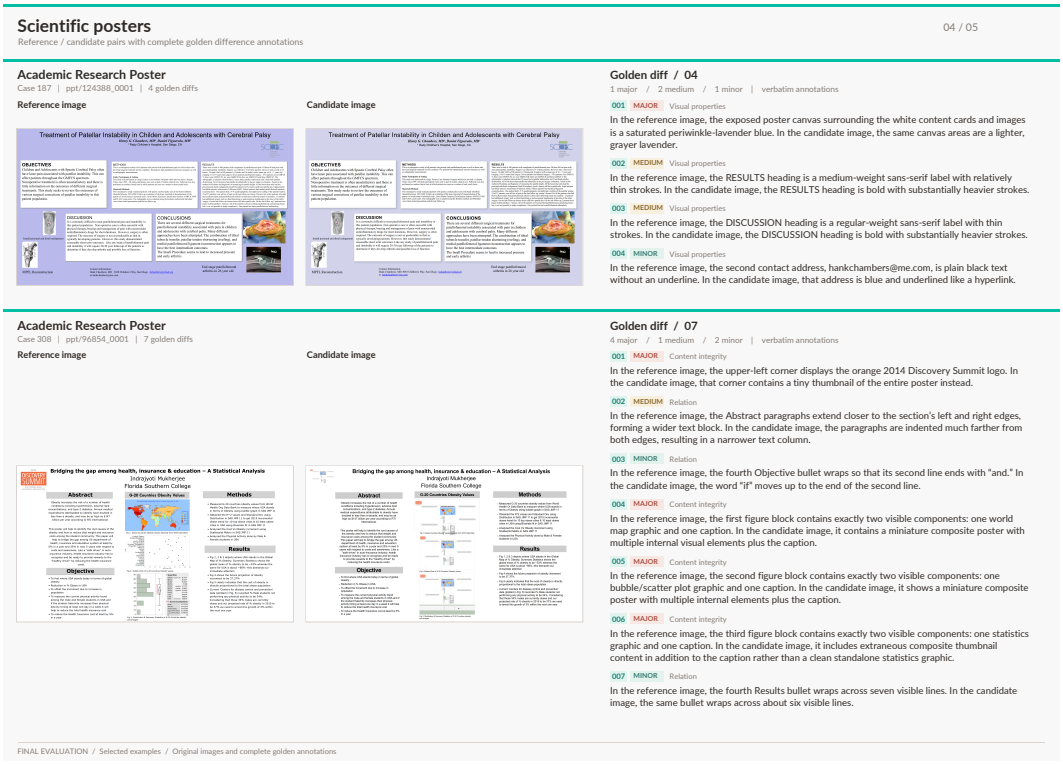

Figure 19: **Qualitative examples of Scientific Posters in DiffBench.**

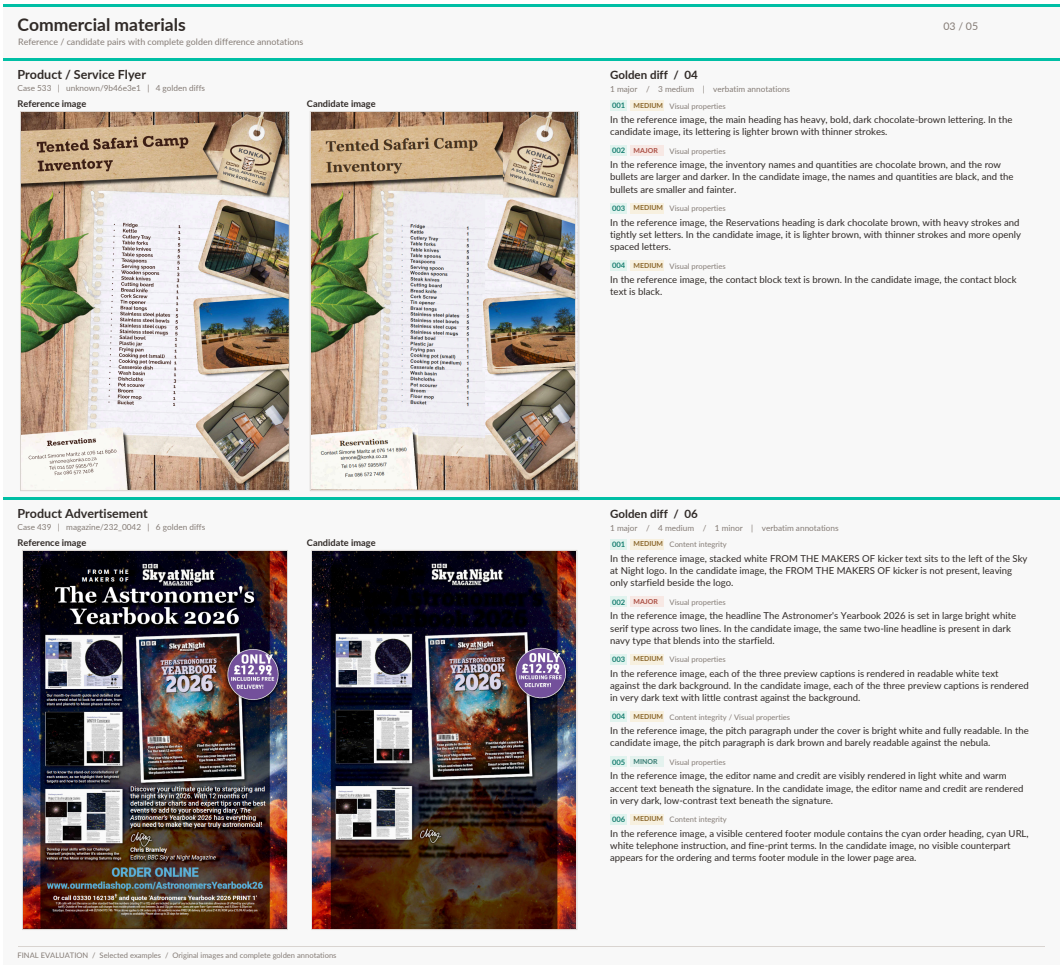

Figure 20: **Qualitative examples of Commercial Materials in DiffBench.**

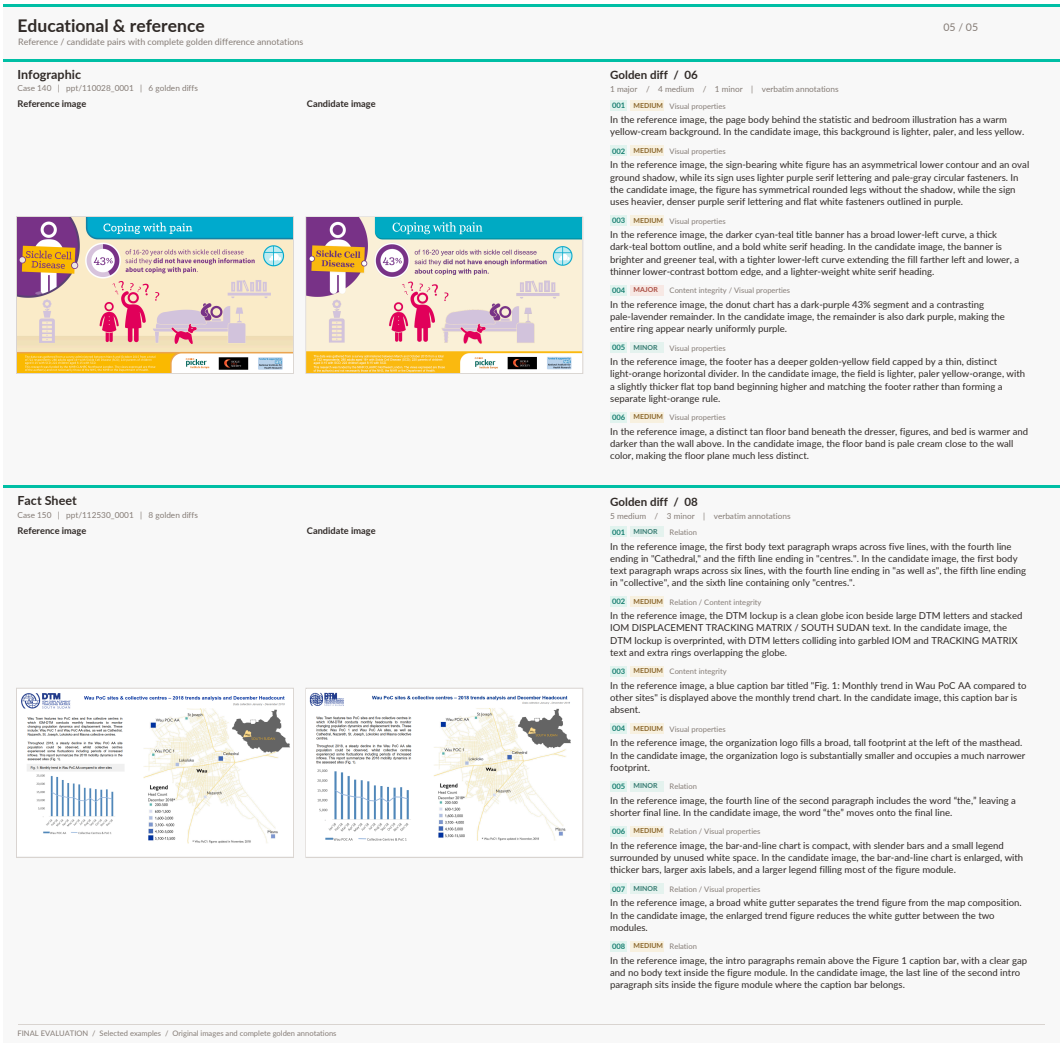

Figure 21: **Qualitative examples of Educational Resources in DiffBench.**