IMAGEPARSER: HIERARCHICAL, GROUNDED IMAGE PARSING WITH SMALL VISION–LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

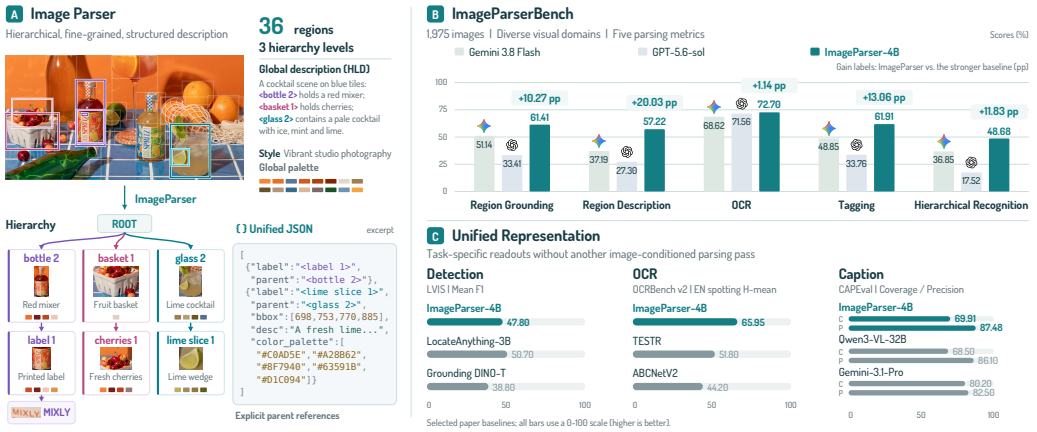

Figure 1: **Illustrating ImageParser.** (A) ImageParser converts an image into a JSON containing global descriptions, style, palettes, and a grounded region hierarchy with parent references. (B) Comparison with frontier LLMs on ImageParserBench. (C) Task-specific readouts derived from the same JSON, predicted for the input image, directly enable high-quality object detection, text spotting, and image captioning.

We investigate *image perception as reconstruction in language*: preserving the descriptive information needed to reconstruct an image up to semantic equivalence. *ImageParser*, a family of 0.8B, 2B, and 4B vision–language models, converts images into hierarchical, spatially grounded JSON parses. Each parse links global content and appearance to localized entities and parts, preserving visible text and explicit parent relationships without predefined categories or downstream questions. We train ImageParser on 6.1 million image–parse pairs produced by frontier perception models and iterative multimodal verification, averaging over 3,000 annotation tokens per image. On *ImageParserBench*, a human-reviewed benchmark of 1,975 images, ImageParser-4B leads all evaluated general-purpose VLMs, including Gemini 3.8 Flash and GPT-5.6-Sol, across six parsing metrics. It improves caption and hierarchy F1 over their strongest external baselines by 20.0 and 11.8 percentage points, respectively. Even the 0.8B model surpasses the evaluated proprietary models in semantics, type, and hierarchy. Task-specific readouts reuse these parses for detection, OCR, and captioning; reconstruction and pixel-free QA assess information preservation and remaining limitations. These results show that compact VLMs learn reusable hierarchical image representations.

1 INTRODUCTION

What would a semantically complete description of an image look like? Consider describing an image to a human or AI system that cannot see it, using as much natural language as needed. Such a description should convey the entities present and their shapes, colors, textures, and materials, together with visible text and the spatial and whole–part relationships that organize the scene. This motivates a fundamental problem in visual perception: how to organize these observations into a coherent, reusable representation that faithfully preserves the image's content, appearance, and structure beyond the requirements of any single downstream task.

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.

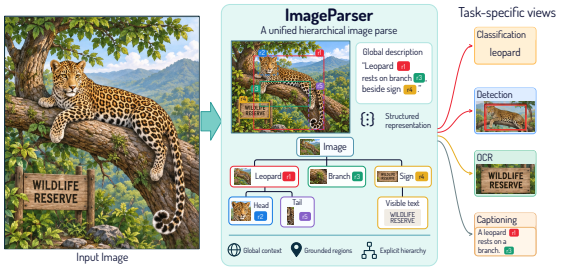

Figure 2: Illustrating the differences between ImageParser and conventional vision tasks. (管中窥豹，可见一斑)

Conventional computer vision has addressed different aspects of this fundamental problem through increasingly expressive, yet deliberately partial, questions of an image. Image classification summarizes an image with category labels (Krizhevsky et al., 2012); object detection identifies entities and their locations (Ren et al., 2015); and segmentation resolves their spatial extent at the pixel level (Kirillov et al., 2019). Optical character recognition recovers visible text, while image captioning verbalizes salient visual content (Li et al., 2023; Garg et al., 2024). Vision–language foundation models support instruction-driven perception (Liu et al., 2023a; Xiao et al., 2024), text-guided localization (Peng et al., 2024), and detailed region captioning (Lian et al., 2025). These tasks have driven remarkable progress in the past few decades, but each specifies in advance which aspect should be extracted. *Each views an image through a narrow, task-specific lens, capturing only the information that its output representation is designed to express.*

To address this challenge, we investigate *image perception as reconstruction in language*. We aim to represent an image's entities, visual attributes, visible text, and spatial and whole–part relationships in structured, spatially grounded language. The goal is to preserve enough information to reconstruct an image up to semantic equivalence: recovering its content, appearance, and composition without requiring identical pixel values. We assess how different models perform on this goal through two complementary evaluations including image reconstruction and pixel-free QA. Image reconstruction tests how faithfully a downstream generator can reproduce these properties from the description, while pixel-free QA tests whether a language model can answer questions about the original image using the description alone. Together, these evaluations probe the information preserved by the representation and its usefulness for generation and reasoning.

We formulate this prediction task as *hierarchical grounded image parsing*. Previous works have provided important steps in this direction: Panoptic Captioning (Lin et al., 2025) pursues image–text equivalence through comprehensive descriptions with explicit locations, while BACON (Yang et al., 2025) and Graph-Based Captioning (Hsieh et al., 2024) organize descriptions into structured representations. Building on these insights, our representation combines a global image description, style and background descriptions, and a set of localized regions. Each region includes a unique identifier, a region type, a bounding box, a parent reference, and a detailed natural-language description, with verbatim text and regional color palettes included where applicable. Together, these components connect the image's overall content and appearance to its spatial and compositional details. The resulting parse is serialized as JSON for human readability and machine processing.

To supervise this task at scale, we sample 6 million images from a pool of more than 4 billion candidates using a uniform sampling strategy based on an internal clustering scheme. We then construct high-quality image–parse pairs through an annotation pipeline that combines frontier vision–language models with specialist models for tasks such as OCR (Li et al., 2026; Tencent Hunyuan Vision Team, 2025) and human pose estimation foundation model Khirodkar et al. (2024). Our pipeline discovers regions, extracts and reconciles visible text, refines descriptions, and constructs and verifies hierarchical relationships through iterative multimodal checks. The resulting annotations average more than 3,000 text tokens per image, providing a detailed and hierarchical supervision for visual content, regional appearance, spatial composition, and whole–part relationships.

Using these annotations, we train *ImageParser*, a family of compact vision–language models with 0.8B, 2B, and 4B parameters. The 4B variant is initialized from Qwen3.5 (Qwen Team, 2026) and trained through full-parameter supervised fine-tuning. Given an image, ImageParser directly generates the structured representation without requiring predefined object categories, a user-selected region, or a downstream question. Classification, detection, OCR, captioning, and grounding can therefore be viewed as *partial projections of a unified image parse*, each drawing on a subset of its semantic, spatial, textual, or relational content (Fig. 1). This allows the same representation to be reused across tasks rather than constructing a separate description for each request.

To evaluate parsing accuracy, we introduce *ImageParserBench*, a human-reviewed benchmark containing 1,975 images and 63,376 annotated regions spanning photographs, graphics, designs, and

artwork. The benchmark measures localization, region semantics, text recognition, region type, and hierarchy. We further evaluate task-specific readouts for detection, OCR, and captioning to examine reuse beyond hierarchical parsing. Following the information-preservation criteria described above, image reconstruction and pixel-free question answering assess the representation's usefulness for generation and reasoning, as well as its remaining limitations (Section 6.4).

On *ImageParserBench*, ImageParser-4B outperforms all evaluated general-purpose VLMs, including Gemini-3.8-Flash, and GPT-5.6-Sol, across all five parsing dimensions. Relative to the strongest external baseline for each metric, it improves caption F1 and hierarchy F1 by 20.0 and 11.8 percentage points, respectively. Even ImageParser-0.8B surpasses the evaluated proprietary models in region semantics, region type, and hierarchy. These results highlight a counterintuitive finding: despite the richness and structural complexity of the target representation, models with just 0.8B–4B parameters can learn to generate accurate, reusable image parses from 6 million high-quality training pairs. These findings suggest that detailed, consistent supervision can enable compact models to perform comprehensive hierarchical grounded image parsing without requiring frontier-scale model capacity. Our downstream evaluations further characterize how effectively the resulting representations support visual understanding and generation.

2 Related Work

Image description and structured visual representations. Detailed descriptions (Krause et al., 2017; Urbanek et al., 2024; Onoe et al., 2024; Garg et al., 2024), spatial grounding (Johnson et al., 2016; Pont-Tuset et al., 2020; Wu et al., 2024; Dwibedi et al., 2024; Hua et al., 2025; Lian et al., 2025), and panoptic coverage (Deng et al., 2025) enrich image captions. Scene graphs organize entities and relations (Krishna et al., 2017; Yang et al., 2022). BACON encodes descriptions as JSON (Yang et al., 2025), Graph-Based Captioning connects grounded region captions (Hsieh et al., 2024), and Panoptic Captioning pursues a minimal textual equivalent of an image (Lin et al., 2025). Building on these works, our parse jointly represents global content and appearance, regions at multiple granularities, verbatim text, and whole–part hierarchy.

Generalist and grounded vision–language models. Generalist VLMs support diverse instructed tasks (Li et al., 2023; Liu et al., 2023a). Kosmos-2, GLaMM, and Florence-2 integrate language with spatial outputs (Peng et al., 2024; Rasheed et al., 2024; Xiao et al., 2024). Their outputs typically depend on the task or query. We instead predict one reusable image parse before downstream questions are known, supporting task-specific readouts, reconstruction, and reasoning.

3 Approach

3.1 Task Formulation: Hierarchical Grounded Image Parsing

Hierarchical grounded image parsing represents image content and composition in language. Given an image I, parser f_θ predicts a structured surrogate without predefined categories, queries, or selected regions:

$$\mathcal{P} = f_\theta(I) = \big(g, \mathcal{R}, \pi\big). \tag{1}$$

Here, g captures global content and appearance, and $\mathcal{R} = \{r_i\}_{i=1}^{N}$ contains regions at multiple granularities. Each $r_i = (b_i, s_i)$ pairs a bounding box with a semantic record comprising an open-vocabulary tag, a detailed description, and visible text where applicable. The parent map $\pi : \{1, \ldots, N\} \to \{0, \ldots, N\}$ assigns each region a parent, with 0 denoting the whole-image root. All parent chains must reach this root, forming a connected, acyclic hierarchy. We serialize regions with explicit parent references as a flat JSON list for autoregressive prediction. As a reusable image surrogate, the parse should faithfully capture the source and support downstream use. We therefore assess *fidelity*—the accuracy and completeness of semantic, spatial, textual, and compositional information—and *sufficiency*—the extent to which the parse supports image reconstruction and question answering without source pixels.

4 Data Curation and Benchmark

Existing datasets provide partial supervision for image description, grounding, or structured understanding, but do not jointly cover the global information, multi-granularity regions, verbatim text,

and whole–part hierarchy required by our task. To support training and evaluation, we develop ParseEngine to generate unified structured annotations and construct 6.1M image–JSON training pairs. We further curate ImageParserBench with human-reviewed references and a unified evaluation protocol for hierarchical grounded image parsing.

4.1 PARSEENGINE

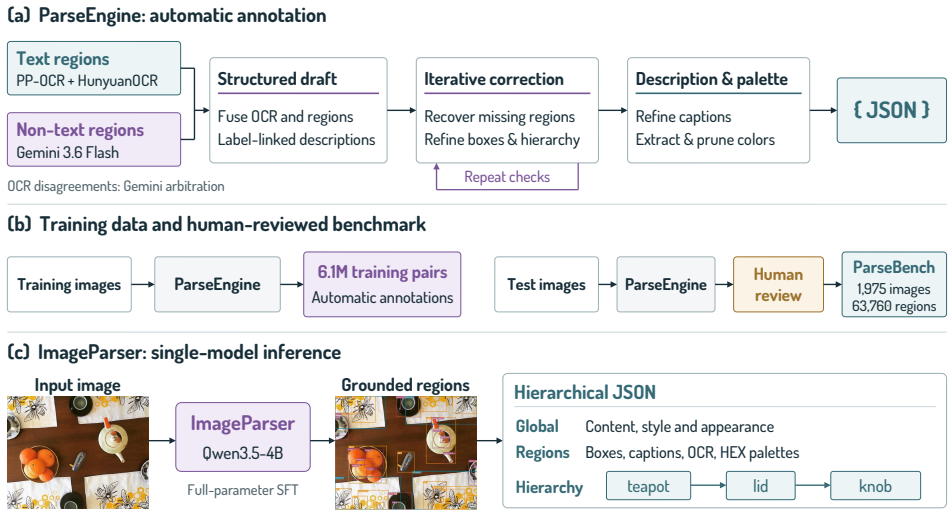

Figure 3: ParseEngine produces training annotations using specialist perception models and multi-modal refinement; benchmark references undergo additional human review.

ParseEngine combines specialist perception models, a commercial VLM, and deterministic validation to produce hierarchical annotations (Figure 3). **(1) Discovery and assembly.** Gemini 3.6 Flash discovers non-text regions with boxes, types, and parents, while PP-OCR and HunyuanOCR extract text. We assemble these outputs with global and region descriptions into a draft. **(2) Validation and refinement.** Rules normalize boxes and repair hierarchy; Gemini iteratively corrects the draft against the image, recovering omissions and removing unsupported or duplicate content. Structural validation follows each update, and a final language pass refines descriptions with geometry, hierarchy, and transcriptions fixed. **(3) Palette extraction.** A pixel-based extractor proposes global and regional colors. Gemini prunes background colors from region palettes and adds color terms to descriptions, selecting only extracted colors while preserving layout and text. Refer to Appendix B for annotation details and examples.

4.2 BENCHMARK CONSTRUCTION AND STATISTICS

ImageParserBench jointly evaluates open-vocabulary semantics, multi-granularity localization, region descriptions, verbatim text, and whole–part hierarchy in a human-reviewed benchmark.

Construction. We stratify 2,000 candidates across 20 content subcategories, sampling semantic clusters proportionally with at most four images per cluster (1,559 clusters; 19 source datasets). Known training/development overlaps are removed where stable source identities are available. Fourteen rounds of human- and VLM-assisted curation check safety, relevance, and category consistency, excluding unsafe, disturbing, and effectively text-only content. RGB standardization and a final manual audit yield **1,975 images**. Reference parses from a fixed ParseEngine configuration undergo additional human review before both annotations and benchmark membership are frozen. These model-assisted references are not exhaustive human-authored ground truth. We retain hashes and pipeline lineage, treat invalid predictions as empty, and exclude extra rows.

Composition and complexity. The benchmark spans 949 photographs, 460 graphics, 405 designs, and 161 artworks (51.9% non-photographic content), with **63,376 regions**. It averages 32.09 regions per image (median 25; 90th percentile 55; maximum 442), with text accounting for 41.36% of regions. Graphics average 47.73 regions versus 24.38 for photographs; tables are densest at 80.86. There are 44,626 direct non-root parent–child edges; maximum depth has a median of two and reaches five, with

top-level regions at depth zero. Every region has a non-empty description, averaging 48.75 words. Figure 4 summarizes these statistics; representative images appear in Appendix A.2 (Figure 9).

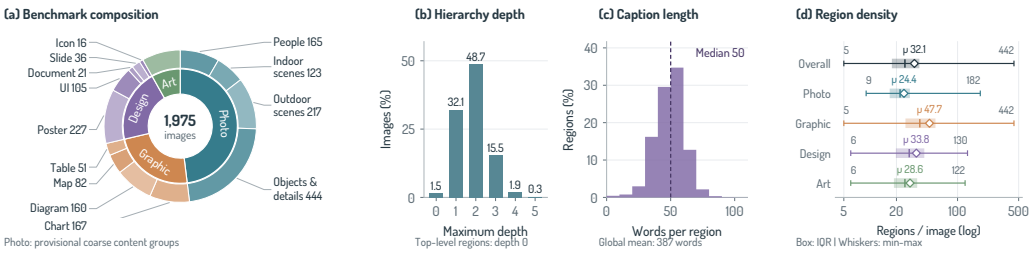

Figure 4: ImageParserBench statistics: (a) domain composition (b) Maximum hierarchy depth (top level: zero). (c) Description length (ten-word bins). (d) Region counts (log scale): boxes indicate quartiles, lines medians, diamonds means, and whiskers extrema.

4.3 Image Parsing Evaluation Protocol

We compare predicted parses with frozen references using six complementary metrics for localization, descriptions, text, types, hierarchy, and color. All scores lie in $[0, 1]$, with higher values indicating better agreement. Refer Appendix E for more details.

Localization and region descriptions. We match predicted and reference regions by box overlap using Hungarian assignment with an IoU threshold of 0.5. Let M be the set of matched region pairs, and let $N_{\mathrm{p}}, N_{\mathrm{g}}$ denote the predicted and reference region counts. *RegionPQ* measures localization, while *Caption F1* measures region-description agreement under geometric matching:

$$\mathrm{RegionPQ} = \frac{2\sum_{(i,j)\in M} \mathrm{IoU}_{ij}}{N_{\mathrm{p}} + N_{\mathrm{g}}}, \qquad \mathrm{F1}_{\mathrm{cap}} = \frac{2\sum_{(i,j)\in M} s_{ij}^{\mathrm{cap}}}{N_{\mathrm{p}} + N_{\mathrm{g}}}. \tag{2}$$

s_{ij}^{cap} is caption cosine similarity clipped to $[0, 1]$; both metrics penalize unmatched regions.

Text recognition. *OCR F1* measures how well visible text is localized and transcribed: $\mathrm{F1}_{\mathrm{ocr}} = \frac{2\sum_{(i,j)\in M_{\mathrm{t}}} s_{ij}^{\mathrm{text}}}{T_{\mathrm{p}}+T_{\mathrm{g}}}$, where M_{t} is the set of matched text-unit pairs, $T_{\mathrm{p}}, T_{\mathrm{g}}$ count predicted and reference text units, and s_{ij}^{text} is normalized character-edit similarity.

Region type and hierarchy. *Type F1* measures macro-averaged region-type correctness: $\mathrm{F1}_{\mathrm{type}} = \frac{1}{C}\sum_{c=1}^{C} \frac{2\mathrm{TP}_c}{2\mathrm{TP}_c+\mathrm{FP}_c+\mathrm{FN}_c}$, with $C = 6$. A true positive requires both a geometric match and the correct type. *Hierarchy F1* measures direct parent–child relations: $\mathrm{F1}_{\mathrm{hier}} = \frac{2\mathrm{TP}_{\mathrm{e}}}{|E_{\mathrm{p}}|+|E_{\mathrm{g}}|}$, where $E_{\mathrm{p}}, E_{\mathrm{g}}$ are predicted and reference parent–child edge sets, excluding links to the root, and TP_{e} counts predicted edges whose endpoints map to a reference edge under M.

Global color palette. *Palette F1@10* measures color agreement and coverage within a fixed tolerance. Let $\mathcal{C}_{\mathrm{p}}, \mathcal{C}_{\mathrm{g}}$ be the predicted and reference color sets, and let M_{10} be a maximum-cardinality one-to-one matching with CIEDE2000 color difference $\Delta E_{00} \le 10$ for every matched pair. We compute $\mathrm{F1}_{\mathrm{palette}}(10) = \frac{2|M_{10}|}{|\mathcal{C}_{\mathrm{p}}|+|\mathcal{C}_{\mathrm{g}}|}$. Missing, extra, and out-of-threshold colors reduce the score.

5 ImageParser

With the developed ParseEngine, we then train a *ImageParser*, the MLLM fine-tuned from Qwen3.5-4B (Qwen Team, 2026) with the annotated image-JSON pairs. Given image I and instruction q, ImageParser predict $y = \mathrm{Serialize}(\mathcal{P})$ with the standard next-token-prediction (NTP) supervison:

$$\mathcal{L}_{\mathrm{NTP}} = -\sum_{t=1}^{|y|} \log p_\theta(y_t \mid I, q, y_{<t}). \tag{3}$$

This objective jointly supervises boxes, descriptions, text, palettes, and parent-child relations.

We employ a two-stage *short-start* recipe to balance structural fidelity and descriptive detail. The first stage learns complete hierarchical parses with concise global and region descriptions. Starting from this checkpoint, the second stage expands the descriptions and introduces *mixed-order supervision*: global-first and layout-first orders are sampled equally for full parses, alongside auxiliary layout-focused targets. This continuation adds detail while preserving structural supervision.

Global-first places image-level descriptions before regions; layout-first places the background and complete region records before global content and style. Both preserve all region fields without removing descriptive content. At inference, the trained model support either way to impose decoding to generate a complete JSON without predefined categories or downstream queries.

6 EXPERIMENTS

Implementation details. We train 0.8B, 2B and 4B ImageParser variants on 6.1M ParseEngine image-JSON pairs using full-parameter fine-tuning and the two-stage training recipe described in Section 5. We compare against LLaVA-OneVision-2-8B, InternVL3.5-38B, Qwen3.8-27B, GLM-5.3-Flash, Gemini 3.8 Flash, and GPT-5.6-sol, prompting each to generate the target schema. Deterministic adapters standardize formats without changing predicted content. The supplement provides hyperparameters, decoding settings, prompts, and hardware configurations.

6.1 COMPARISON WITH STATE-OF-THE-ART MODELS ON IMAGEPARSERBENCH

Table 1: Hierarchical parsing on ImageParserBench. Throughout the paper, bold and underlined scores mark the best and second-best results within each group.

Model	RegionPQ ↑	Caption F1 ↑	OCR F1 ↑	Type F1 ↑	Hierarchy F1 ↑	Palette F1@10 ↑
LLaVA-OneVision-2-8B	0.1145	0.0996	0.1211	0.0634	0.0271	–
InternVL3.5-38B	0.3075	0.2323	0.2471	0.2325	0.1414	–
Qwen3.8-27B	0.4337	0.3585	0.5972	0.3762	0.2881	0.3780
GLM-5.3-Flash	0.3154	0.0721	0.4681	0.2941	0.2052	0.4459
Gemini 3.8 Flash	0.5114	0.3719	0.6862	0.4885	0.3685	0.4157
GPT-5.6-Sol	0.3341	0.2730	<u>0.7156</u>	0.3376	0.1752	0.6135
ImageParser-0.8B	0.5046	0.4631	0.6272	0.5332	0.3804	0.8768
ImageParser-2B	<u>0.5629</u>	<u>0.5237</u>	0.6992	<u>0.5717</u>	<u>0.4406</u>	<u>0.8807</u>
ImageParser-4B	**0.6141**	**0.5722**	**0.7270**	**0.6191**	**0.4868**	**0.9009**

Table 1 shows that ImageParser's main advantages lie in discovering and organizing regions and describing them consistently with the target representation. ImageParser-4B improves RegionPQ from 0.5114 to 0.6141 and Caption F1 from 0.3719 to 0.5722 over the strongest baselines, alongside gains in region type and hierarchy. These advantages persist at smaller scales: ImageParser-0.8B already outperforms many external baselines on Caption, Type, and Hierarchy F1. Together, these results suggest that a unified parsing target and structured supervision enable even compact models to recover rich, organized image content.

6.2 ABLATION STUDIES

We conduct ablations to examine how our training recipe and inference-time decoding order affect parsing fidelity and reconstruction quality (Table 2). **(1) Longer descriptions improve captioning and reconstruction.** Compared with short-only training, long-only training improves caption F1 (0.5152 to 0.5482) and all four reconstruction metrics, but slightly lowers RegionPQ (0.5974 to 0.5925), OCR, type, and hierarchy F1. Richer descriptions therefore offer a trade-off rather than uniform parsing gains. **(2) Short-start is the stronger training recipe.** Under layout-first decoding, short-start mixed-order training outperforms long-start on all five parsing and four reconstruction metrics, including RegionPQ (0.6141 vs. 0.5950). This supports continuation from a short-description checkpoint. **(3) Layout-first favors reconstruction quality.** For the same short-start checkpoint, layout-first improves all four reconstruction metrics over global-first, including SSIM (0.5046 to 0.5100). RegionPQ and caption F1 differ by only 0.0002 and 0.0001, respectively; OCR improves slightly, while type and hierarchy F1 decline.

Table 2: Ablations of training recipe and decoding order on ImageParserBench. Global/Layout denote global-first/layout-first decoding. The last two rows use the same checkpoint, with layout-first decoding as our default. Reconstruction uses the fixed in-house renderer.

Training	Decoding	Parsing fidelity					Reconstruction fidelity			
		RegionPQ↑	Caption F1↑	OCR F1↑	Type F1↑	Hierarchy F1↑	SSIM↑	Edge↑	DISTS↓	Color↑
Short only	Global	0.5974	0.5152	0.6917	0.6109	0.4658	0.5009	0.7116	0.1792	0.7496
Long only	Global	0.5925	0.5482	0.6842	0.6040	0.4585	<u>0.5072</u>	0.7214	**0.1704**	**0.7767**
Long-start	Layout	0.5950	0.5533	0.6983	0.6013	0.4592	0.5013	<u>0.7220</u>	0.1739	0.7462
Short-start	Global	**0.6143**	**0.5723**	<u>0.7261</u>	**0.6247**	**0.4964**	0.5046	0.7202	0.1722	0.7642
Short-start	Layout	<u>0.6141</u>	<u>0.5722</u>	**0.7270**	<u>0.6191</u>	<u>0.4868</u>	**0.5100**	**0.7244**	<u>0.1708</u>	<u>0.7664</u>

6.3 Task-Specific Projections of the Unified Parse

We test whether a single unified parse can support multiple perception tasks through simple readouts. Each image is parsed once with a task-independent instruction and no task-specific fine-tuning. Outputs are obtained through language-model selection or direct field extraction from the frozen JSON. Task-specific queries are introduced only at readout, without rerunning ImageParser or revisiting source pixels (Table 3). **(1) Object detection.** A text-only Gemini 3.8 Flash adapter selects existing boxes from region descriptions without changing their coordinates. Following LocateAnything's per-image GT-positive category queries and FastEvaluate protocol (Wang et al., 2026) (Appendix F.1), this readout achieves 47.8 mean F1, exceeding all listed baselines except LocateAnything (50.7); its F1@0.95 of 25.1 also ranks second. **(2) Text spotting.** We extract transcription–box pairs from the unified parse and adapt their word/line granularity to each source dataset (Appendix F.2). This readout achieves 65.95 H-mean on OCRBench v2's public English Text Spotting subset (Fu et al., 2025), surpassing all listed baselines, including the specialist TESTR (51.80). **(3) Caption-content readout.** Gemini 3.8 Flash evaluates the complete JSON against CAPEval's semantic checklists (Liu et al., 2026), without additional caption generation. This yields 69.91 Coverage and 87.48 Precision, numerically exceeding all listed baselines in Precision and all listed open-source baselines in Coverage (Appendix F.3). Published caption-generation baselines provide reference values under a different protocol. Together, these readouts show that the unified representation preserves spatial, textual, and descriptive information for downstream use.

Table 3: Detection, text spotting, and captioning (%, ↑) performance probe from the unified parse. CAPEval uses a different judge protocol from the published caption baselines.

(a) LVIS detection

Method	F1@0.50	F1@0.95	Mean F1
Grounding DINO-T	47.7	22.7	38.8
DeepSeek-VL2-Small	56.2	21.0	41.8
Qwen3-VL-4B	59.8	20.0	43.5
Qwen3-VL-8B	61.5	20.2	44.8
Cosmos-Reason2-8B	56.4	9.8	40.2
SEED1.5-VL	**65.6**	19.5	46.7
Rex-Omni-3B	<u>64.3</u>	20.7	46.9
LocateAnything-3B	62.3	**31.1**	**50.7**
ImageParser-4B	61.1	<u>25.1</u>	<u>47.8</u>

(b) OCRBench v2

Method	H-mean ↑
ABCNet	32.20
ABCNetV2	44.20
TESTR	<u>51.80</u>
GPT-4o	0.00
Gemini-1.5-Pro	13.50
Step-1V	7.20
InternVL3-14B	11.20
Qwen2.5-VL-7B†	51.60
ImageParser-4B	**65.95**

(c) CAPEval

Method	Coverage ↑	Precision ↑
Gemini-3.1-Pro	**80.2**	82.5
Gemini-3.5-Flash	<u>78.1</u>	81.4
Gemini-2.5-Pro	75.5	80.1
GPT-5.5	72.8	79.0
Qwen3-VL-32B	68.5	<u>86.1</u>
GLM-4.6V-Flash	62.0	85.4
InternVL3.5-38B	48.6	77.7
LLaVA-OV-1.5-8B	41.2	50.4
ImageParser-4B	69.91	**87.48**

6.4 Downstream Application of the Image Parser

The parsing metrics above measure agreement with reference annotations. We next ask whether the parse preserves enough information to serve as a useful, pixel-free surrogate for the source image. We evaluate this through two complementary tasks: JSON-to-image reconstruction and pixel-free Question Answering, with downstream models receiving the parse rather than source pixels.

JSON-to-Image Reconstruction. We test whether the unified parse preserves enough information to reconstruct an image (Figure 5). Given I, ImageParser produces $\hat{\mathcal{P}} = f_\theta(I)$, which a renderer converts into $\tilde{I} = G(\phi_G(\hat{\mathcal{P}}))$ using a deterministic schema adapter ϕ_G, without source pixels or visual features. We compare parsers on ImageParserBench using a two different JSON-to-Image render including an in-house image renderer and an external Ideogram 4 (Ideogram AI, 2026). We report mean scores over source–reconstruction pairs: SSIM and edge similarity measure structure, DISTS measures perceptual differences, OCR F1 measures text retention, and color similarity measures

global color fidelity. Our JSON achieves the best reported results across all five metrics with both renderers (Table 4), supporting its utility across different generators.

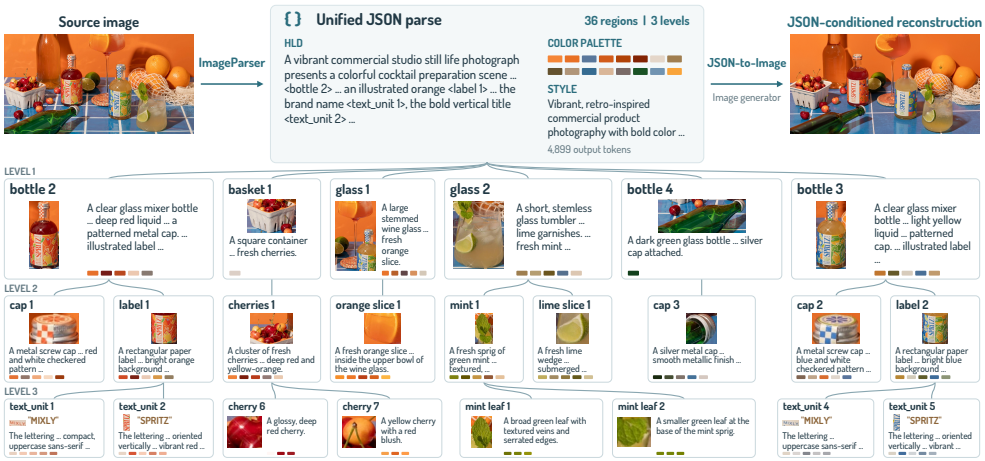

Figure 5: **Hierarchical JSON-to-image reconstruction.** Branches show object–part–text structure; rendering uses the complete JSON without pixels. Further examples: Appendix I.

Figure 6 illustrates improvements in local detail and overall color: compared with GPT-5.6-sol JSON under the same Ideogram renderer, our image parser better preserves the bedside dish's shallow shape and reduces the scene's warm color shift. The in-house reconstruction further matches the dish's white appearance and the source image's neutral tones. Refer to Appendix G for preprocessing and implementation details.

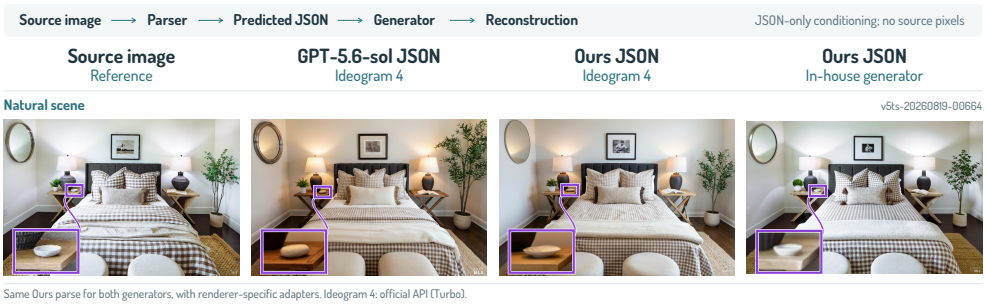

Figure 6: **JSON-conditioned reconstruction.** From left: source image, GPT-5.6-sol JSON with Ideogram 4, and our JSON with Ideogram 4 and the in-house renderer.

Table 4: ImageParserBench reconstruction with the in-house renderer and Ideogram 4.

	In-house JSON-to-Image					External JSON-to-Image				
Model	SSIM ↑	Edge Structure ↑	DISTS ↓	OCR F1 ↑	Color ↑	SSIM ↑	Edge Structure ↑	DISTS ↓	OCR F1 ↑	Color ↑
Qwen3.8-27B	0.3461	0.6190	0.2398	0.3574	0.5414	0.3529	0.6300	0.2340	0.3782	0.5799
GLM-5.3-Flash	0.3406	0.6124	0.2303	0.2804	0.5776	0.3375	0.6246	0.2287	0.3009	0.6015
Gemini-3.8-Flash	0.3828	0.6362	0.2289	0.3922	0.5748	0.3827	0.6461	0.2282	0.3809	0.6109
GPT-5.6-Sol	0.4145	0.7070	0.2004	0.4309	0.6322	0.3868	0.6474	0.2114	0.3751	0.6279
Ours (layout-first)	**0.5100**	**0.7244**	**0.1708**	**0.4773**	**0.7664**	**0.4040**	**0.6536**	**0.2101**	**0.4037**	**0.6989**

Pixel-Free Question Answering. We test whether a question-independent parse can serve as reusable visual evidence. ImageParser runs once per image before observing any downstream question; a QA reader receives only the frozen JSON and the question, without pixels or visual features. We compare this readout with direct image input to the same reader on five external benchmarks, independent of ImageParserBench: MMStar (Chen et al., 2024) and MMBench v1.1 (Liu et al., 2024) for general perception and reasoning, ChartQA (Masry et al., 2022) for chart understanding, OCRBench v1 (Liu et al., 2023b) for text-centric understanding, and InfographicVQA (Mathew et al., 2022) for composed informational layouts. We fix ImageParser-4B and use Qwen3-VL-8B-

Instruct (Bai et al., 2025) and Gemini 3.8 Flash as readers. Gemini also answers from its own question-independent parse (Self-JSON). Appendix H details the evaluation protocol.

Table 5: Pixel-Free QA (%, ↑). Our JSON: ImageParser-4B; Self-JSON: Gemini's own parse. Δ: JSON minus image-only (pp); darker shading indicates larger deficits.

	Qwen3-VL-8B			Gemini 3.8 Flash			Gemini 3.8 Flash		
Benchmark	Image-only	Our JSON	Δ	Image-only	Our JSON	Δ	Image-only	Self-JSON	Δ
MMStar	72.47	53.80	−18.67	84.33	78.07	−6.27	84.33	79.07	−5.27
MMBench v1.1	75.93	70.90	−5.03	94.74	90.17	−4.57	94.74	93.27	−1.47
ChartQA	85.44	71.48	−13.96	91.48	89.24	−2.24	91.48	84.48	−7.00
OCRBench v1	85.30	68.80	−16.50	90.40	76.60	−13.80	90.40	84.70	−5.70
InfographicVQA	83.99	67.05	−16.94	92.68	76.99	−15.69	92.68	60.09	−32.59

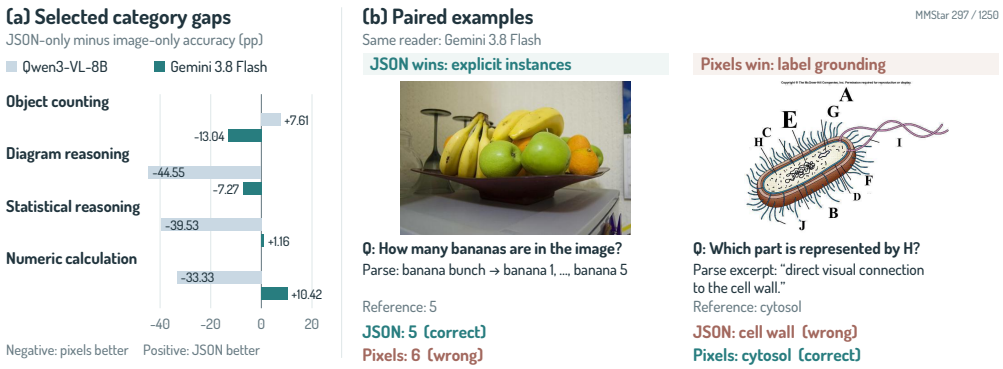

Figure 7: **MMStar analysis.** (a) JSON-minus-pixel accuracy gaps. (b) Paired Gemini cases: JSON correctly counts bananas; pixels correctly identify label H. Images are shown for inspection only; excerpts are shortened. Full analysis: Appendix H.

Table 5 shows that direct image input remains stronger overall, but ImageParser JSON supports substantial QA performance: with Gemini, it reaches 89.24 on ChartQA and 90.17 on MMBench, within 2.24 and 4.57 percentage points of the corresponding image-only scores. We further find that *(1) the same parse supports substantially better QA with Gemini than with Qwen.* Gemini raises JSON-only scores by 24.27 points on MMStar and 17.76 on ChartQA, despite receiving the same ImageParser outputs. *(2) ImageParser JSON is more effective than Self-JSON on charts and infographics.* With the Gemini reader fixed, ImageParser JSON leads by 4.76 points on ChartQA and 16.90 on InfographicVQA, while Self-JSON performs better on the other three benchmarks. *(3) JSON's advantage over pixels can reverse across readers.* On MMStar, JSON improves counting by 7.61 points with Qwen but trails pixels by 13.04 with Gemini; statistical reasoning shows the opposite pattern, shifting from a 39.53-point deficit to a 1.16-point gain (Figure 7(a)). Thus, a deficit with one reader need not indicate that the parse lacks the required evidence. The paired Gemini examples in Figure 7(b) illustrate how evidence quality also matters: explicit banana instances support a correct count, while an incorrect label-to-part description accompanies a wrong diagram answer.

7 Conclusion

We studied *image perception as reconstruction in language* through hierarchical grounded image parsing, integrating global content and appearance, localized entities and parts, visible text, and explicit parent relationships into a reusable representation. Trained on 6.1 million image–parse pairs, our *ImageParser* models with 0.8B–4B parameters learn detailed structured descriptions, with the 4B model outperforming all evaluated frontier general-purpose VLMs (e.g., *Gemini-3.8-Flash, GPT-5.6-Sol*) on ImageParserBench. These results show that rich hierarchical parsing can be learned by compact models with detailed supervision. Task-specific readouts demonstrate reuse for detection, OCR, and captioning, while reconstruction and pixel-free QA probe information preservation and its limits. In particular, pixel-free QA highlights the importance of both retaining visual evidence and enabling readers to use it effectively. Together, these findings support structured language as a practical interface connecting visual perception with downstream generation and reasoning.

AI USE STATEMENT

In the course of this work, Large Language Models (LLMs) were used as auxiliary tools to support the research process. Specifically, we employed LLMs as a sounding board during the design of experiments and the analysis of results, using them to challenge our reasoning and to help identify additional comparisons and controls worth considering. For literature search, we used LLMs equipped with web search to surface potentially relevant prior work; all sources were subsequently read and verified by the authors, and the descriptions and interpretations of prior work are entirely our own. We also relied on LLMs for language editing, including improving clarity and readability, correcting grammatical errors, and refining sentence phrasing and overall text flow.

The authors are fully responsible for all content in this manuscript, including any text produced or edited with the assistance of LLMs. We have taken care to ensure that such assistance complies with ethical standards and does not give rise to plagiarism or any form of scientific misconduct.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models? In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2403.20330.

Xueqing Deng, Linjie Yang, Qihang Yu, Ali Athar, Chenglin Yang, Xiaojie Jin, Xiaohui Shen, and Liang-Chieh Chen. COCONut-PanCap: Joint panoptic segmentation and grounded captions for fine-grained understanding and generation. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Debidatta Dwibedi, Vidhi Jain, Jonathan Tompson, Andrew Zisserman, and Yusuf Aytar. FlexCap: Describe anything in images in controllable detail. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Ling Fu, Zhebin Kuang, Jiajun Song, Mingxin Huang, Biao Yang, Yuzhe Li, Linghao Zhu, Qidi Luo, Xinyu Wang, Hao Lu, Zhang Li, Guozhi Tang, Bin Shan, Chunhui Lin, Qi Liu, Binghong Wu, Hao Feng, Hao Liu, Can Huang, Jingqun Tang, Wei Chen, Lianwen Jin, Yuliang Liu, and Xiang Bai. OCRBench v2: An improved benchmark for evaluating large multimodal models on visual text localization and reasoning. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://arxiv.org/abs/2501.00321.

Roopal Garg, Andrea Burns, Burcu Karagol Ayan, Yonatan Bitton, Ceslee Montgomery, Yasumasa Onoe, Andrew Bunner, Ranjay Krishna, Jason Baldridge, and Radu Soricut. ImageInWords: Unlocking hyper-detailed image descriptions. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Yu-Guan Hsieh, Cheng-Yu Hsieh, Shih-Ying Yeh, Louis Béthune, Hadi Pour Ansari, Pavan Kumar Anasosalu Vasu, Chun-Liang Li, Ranjay Krishna, Oncel Tuzel, and Marco Cuturi. Graph-based captioning: Enhancing visual descriptions by interconnecting region captions. *arXiv preprint arXiv:2407.06723*, 2024.

Hang Hua, Qing Liu, Lingzhi Zhang, Jing Shi, Soo Ye Kim, Zhifei Zhang, Yilin Wang, Jianming Zhang, Zhe Lin, and Jiebo Luo. FINECAPTION: Compositional image captioning focusing on wherever you want at any granularity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24763–24773, 2025.

Ideogram AI. Ideogram 4. https://ideogram.ai/models/4.0, 2026.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. DenseCap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4565–4574, 2016.

Rawal Khirodkar, Timur Bagautdinov, Julieta Martinez, Su Zhaoen, Austin James, Peter Selednik, Stuart Anderson, and Shunsuke Saito. Sapiens: Foundation for human vision models. In *ECCV*, pp. 206–228. Springer, 2024. doi: 10.1007/978-3-031-73235-5_12.

Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9404–9413, 2019.

Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. A hierarchical approach for generating descriptive image paragraphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 317–325, 2017.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73, 2017. doi: 10.1007/s11263-016-0981-7.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, pp. 1097–1105, 2012.

Gengluo Li, Xingyu Wan, Shangpin Peng, Weinong Wang, Hao Feng, Yongkun Du, Binghong Wu, Zheng Ruan, Zhiqiong Lu, Liang Wu, Pengyuan Lyu, Huawen Shen, Zibin Lin, Shijing Hu, Jieneng Yang, Hongbing Wen, Guanghua Yu, Hong Liu, Bochao Wang, Can Ma, Han Hu, Chengquan Zhang, and Yu Zhou. HunyuanOCR-1.5: Making lightweight OCR VLMs faster and better. *arXiv preprint arXiv:2607.04884*, 2026.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning (ICML)*, 2023.

Long Lian, Yifan Ding, Yunhao Ge, Sifei Liu, Hanzi Mao, Boyi Li, Marco Pavone, Ming-Yu Liu, Trevor Darrell, Adam Yala, and Yin Cui. Describe anything: Detailed localized image and video captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 21766–21777, 2025.

Kun-Yu Lin, Hongjun Wang, Weining Ren, and Kai Han. Panoptic captioning: An equivalence bridge for image and text. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 216–233, 2024.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xucheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. OCRBench: On the hidden mystery of OCR in large multimodal models. *arXiv preprint arXiv:2305.07895*, 2023b.

Zhipeng Liu, Haochen Wang, and Zhaoxiang Zhang. CAPEval: A decoupled caption evaluation across understanding and generation. *arXiv preprint arXiv:2608.02589*, 2026. URL https://arxiv.org/abs/2608.02589.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2263–2279. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.177.

Minesh Mathew, Viraj Bagal, Ruben Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V. Jawahar. InfographicVQA. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1697–1706, 2022. URL https://openaccess.thecvf.com/content/WACV2022/html/Mathew_InfographicVQA_WACV_2022_paper.html.

Yasumasa Onoe, Sunayana Rane, Zachary Berger, Yonatan Bitton, Jaemin Cho, Roopal Garg, Alexander Ku, Zarana Parekh, Jordi Pont-Tuset, Garrett Tanzer, Su Wang, and Jason Baldridge. DOCCI: Descriptions of connected and contrasting images. In *European Conference on Computer Vision (ECCV)*, 2024.

Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, Qixiang Ye, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. In *International Conference on Learning Representations (ICLR)*, 2024.

Jordi Pont-Tuset, Jasper Uijlings, Soravit Changpinyo, Radu Soricut, and Vittorio Ferrari. Connecting vision and language with localized narratives. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Hanoona Rasheed, Muhammad Maaz, Sahal Shaji, Abdelrahman Shaker, Salman Khan, Hisham Cholakkal, Rao M. Anwer, Eric Xing, Ming-Hsuan Yang, and Fahad S. Khan. GLaMM: Pixel grounding large multimodal model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13009–13018, 2024.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

Tencent Hunyuan Vision Team. HunyuanOCR technical report. *arXiv preprint arXiv:2511.19575*, 2025.

Jack Urbanek, Florian Bordes, Pietro Astolfi, Mary Williamson, Vasu Sharma, and Adriana Romero-Soriano. A picture is worth more than 77 text tokens: Evaluating CLIP-style models on dense captions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26700–26709, 2024.

Shihao Wang, Shilong Liu, Yuanguo Kuang, Xinyu Wei, Yangzhou Liu, Zhiqi Li, Yunze Man, Guo Chen, Andrew Tao, Guilin Liu, Jan Kautz, Lei Zhang, and Zhiding Yu. LocateAnything: Fast and high-quality vision-language grounding with parallel box decoding. *arXiv preprint arXiv:2605.27365*, 2026. URL `https://research.nvidia.com/labs/lpr/locate-anything/`.

Jialian Wu, Jianfeng Wang, Zhengyuan Yang, Zhe Gan, Zicheng Liu, Junsong Yuan, and Lijuan Wang. GRiT: A generative region-to-text transformer for object understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 207–224, 2024.

Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4818–4829, 2024.

Jingkang Yang, Yi Zhe Ang, Zujin Guo, Kaiyang Zhou, Wayne Zhang, and Ziwei Liu. Panoptic scene graph generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 178–196, 2022. doi: 10.1007/978-3-031-19812-0_11.

Zhantao Yang, Ruili Feng, Keyu Yan, Huangji Wang, Zhicai Wang, Shangwen Zhu, Han Zhang, Jie Xiao, Pingyu Wu, Kai Zhu, Jixuan Chen, Chen-Wei Xie, Yue Yang, Hongyang Zhang, Yu Liu, and Fan Cheng. BACON: Improving clarity of image captions via bag-of-concept graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14380–14389, 2025.

A Data Sources and Annotation Quality

A.1 Training Corpus and Data Separation

Our training corpus contains 6.1M image–JSON pairs, constructed by randomly sampling images from a collection of approximately 3.65B images and annotating them with ParseEngine. The source collection is deduplicated across data sources using perceptual hashes. SigLIP2 representations organize the collection into 10,000 visual clusters, grouped into six broad categories and 20 subcategories. The training data cover objects and products, scenes and places, people, living things and food, design, and documents (Figure 8(a)). These categories include apparel, furniture, packaging, electronics, vehicles, natural and urban scenes, portraits, food, plants, animals, posters, interfaces, cartoons, artwork, and multilingual documents.

The corpus also spans a broad range of image resolutions and aspect ratios. Approximately 46.1% of images have a longest side of 512–1,023 pixels, and 29.2% have a longest side of at least 1,024 pixels (Figure 8(b)). In terms of width-to-height ratio, approximately 52.9% lie in $[1.25, 2)$, 24.4% in $[0.8, 1.25)$, and 18.8% in $[0.5, 0.8)$, covering landscape, near-square, and portrait formats.



Figure 8: Composition of the 6.1M training corpus by content category and long-side resolution. Percentages are rounded.

The annotation procedure, construction of paired short and long description targets, and data-separation checks are detailed in Appendix B.2–B.4.

A.2 ImageParserBench

ImageParserBench undergoes manual sample selection and reference-parse refinement following Appendix B.2–B.3, before the images and annotations are frozen for evaluation. Figure 9 shows representative Photo, Graphic, Design, and Art images, complementing the composition and complexity statistics in Figure 4.

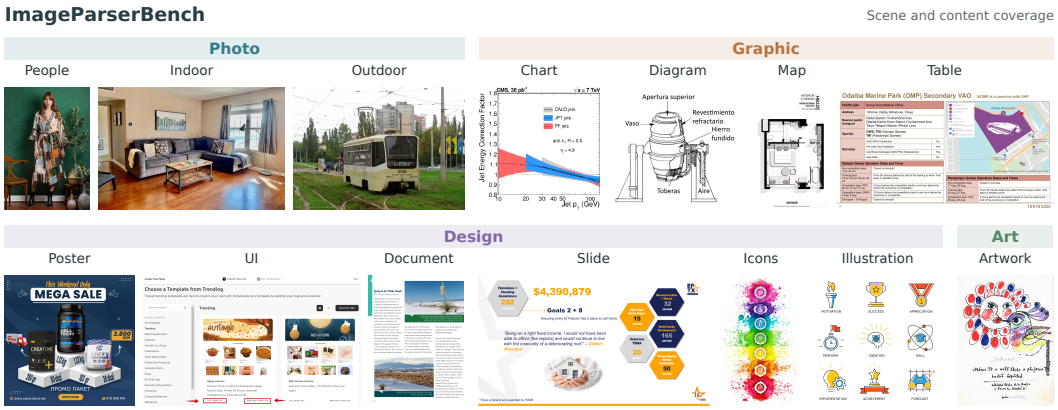

Figure 9: Representative ImageParserBench images grouped into Photo, Graphic, Design, and Art. Colored bands indicate domains and panel labels describe the illustrated content; original aspect ratios are preserved.

B ParseEngine Implementation and Qualitative Example

B.1 Dataset Comparison and Annotation Fields

Table 6 compares annotation fields rather than whether localization was an original prediction target. *Global caption* denotes an image-level description. *Region caption* denotes localized descriptive text, including grounded phrases or passages rather than category labels alone. *Grounding* links text to boxes or masks, and *hierarchy* denotes explicit parent–child or containment links. *OCR annotation* requires a dedicated region-linked transcription field; *color palette* requires structured color codes rather than color words in captions. Image counts refer to the listed datasets or subsets, and our row describes the 6.1M-image training corpus rather than ImageParserBench.

Table 6: Comparison with existing image-caption datasets.

Dataset	Images	Structure		Descriptions		Fine-grained Annotations	
		Grounding	Hierarchy	Global Caption	Region Caption	OCR	Palette
DCI	7.8K	✓	✓	✓	✓	✗	✗
DOCCI	14.8K	✗	✗	✓	✗	✗	✗
IIW-400	400	✓	✗	✓	✓	✗	✗
GCG	56.9K	✓	✗	✓	✓	✗	✗
SA-Pancap	9.6K	✓	✗	✓	✓	✗	✗
GBC10M	10.1M	✓	✓	✓	✓	✗	✗
Ours (training)	**6.1M**	✓	✓	✓	✓	✓	✓

B.2 Detailed Annotation Pipeline

We expand the three-stage overview in Section 4.1 into the operations used to construct an image parse. ParseEngine first establishes a region scaffold, assembles textual and visual evidence, and corrects the resulting draft against the image. It then freezes the structure before producing the final descriptions and palettes. Training annotations are generated automatically; benchmark samples undergo final manual selection and annotation refinement after the automatic pipeline, as described below.

Hierarchical region discovery and OCR. Two complementary branches process the image. Gemini 3.6 Flash proposes a coarse-to-fine region plan with image-local identities, labels, types, bounding boxes, and parent references. This plan identifies entities and parts without committing to final descriptions or transcriptions. OCR experts independently return visible text and text-line boxes. The recorded production configuration uses HunyuanOCR-1.5, while the frozen benchmark configuration uses PP-OCRv6. Both branches are mapped to a common normalized coordinate frame before fusion; the specific OCR configuration is tied to the annotation snapshot.

Structured description and OCR fusion. The VLM receives the image, region plan, and OCR detections to produce initial global, style, background, and region descriptions. It also groups OCR lines into textual units and assigns their parents. A rule-based assembler removes provisional text proposals and replaces them with OCR-derived units. In the recorded fusion implementation, a non-text proposal whose maximum IoU with an OCR box exceeds 0.5 is also removed as a potential duplicate. This geometric heuristic does not establish semantic equivalence. Each assembled text unit takes its box from the union of its selected OCR-line boxes and its transcription from those lines. OCR-line reconciliation recovers unassigned lines and resolves repeated assignments so that each OCR line is represented once at this stage. The retained non-text geometry is preserved during assembly. Transcription is stored in `text`, separately from the description of typography and visual appearance in `desc`.

Canonical cleanup and entity enrichment. Deterministic checks normalize boxes to the $[0, 1000]$ coordinate frame, enforce unique region labels, repair dangling parent references and cycles, and place parents before children in the flat region list. Label changes are propagated to parent references and global-description references. Unresolved parents fall back to a geometrically compatible region or the synthetic root. An optional entity-enrichment step proposes high-confidence names for non-text regions and removes redundant assignments along parent–child chains. Guarded updates preserve

geometry, region types, parent links, and text-region content. These operations establish a well-formed draft; they do not by themselves verify visual correctness. Appendix C describes the serialized representation, schema-version correspondence, and parsing prompts.

Image-aware layout correction. The VLM revisits the image and complete draft to recover missing regions, remove unsupported or duplicate content, adjust boxes, and revise types, parent assignments, and existing transcriptions where needed. Canonical repair follows structural updates. Thus, the OCR provenance and geometry-preservation guarantees of the assembly stage do not prevent later image-aware correction. After layout correction, region identities, boxes, types, parent links, and transcriptions are frozen for the language-refinement stage.

B.3 Short and Long Description Targets

The final language pass generates concise and detailed versions of the global and region descriptions from the same image and corrected parse. Both versions retain the complete region scaffold: short descriptions reduce linguistic detail without removing regions, boxes, parent links, or visible-text transcriptions. The long version expands the descriptive content while referring to the same localized entities and parts.

The pairing therefore follows the source image and the fixed image-local region identities. Each region's short and long descriptions refer to the same box and parent assignment; global descriptions refer to the same parse. No separate region discovery or geometric matching is needed to establish this correspondence. Structure and transcription remain fixed during description refinement, as do existing entity assignments. These paired description variants provide the concise and detailed supervision used by the short-start recipe in Section 5. Description length is distinct from serialization order: global-first and layout-first reorder the fields of a parse without changing the region correspondence.

Palette extraction and semantic pruning. After description refinement, a CPU-based extractor obtains up to 16 colors from the image and five from each region crop. A rectangular crop may include background colors, so the VLM selects the subset that corresponds to the region's subject. Guards permit only hexadecimal colors already present in the extracted palette; if the retained subset is empty, the original palette is kept. Color terms may be updated in non-text descriptions, while text-region descriptions, region identities, geometry, hierarchy, and transcriptions remain unchanged. This stage adds appearance information without changing the correspondence between description variants.

Human selection and refinement for the benchmark. After the automatic annotation pipeline, benchmark candidates undergo manual screening. The generated reference parses of retained samples are reviewed against their source images and refined to correct annotation errors before finalization. The selected images and refined references are then frozen for evaluation.

B.4 Data Separation and Remaining Limitations

Source-collection deduplication and benchmark overlap filtering address different stages of data construction. The source collection described in Appendix A is deduplicated across data sources using perceptual hashes. During benchmark curation, known overlaps with training or development data are removed where stable source identities are available. The latter check operates on image identity, independently of whether its annotation uses short or long descriptions.

Identity-based filtering does not rule out independently ingested near-duplicates with different source identities. Benchmark membership and reviewed references are frozen for evaluation; later illustrative annotations do not replace those references.

B.5 Qualitative Example

Figure 10 uses PP-OCR followed by Gemini refinement. Visual verification corrects an OCR misreading of `CELL TECH` and recovers a missed plus sign; the price-sticker subtree illustrates explicit parent–child links. Additional non-text regions are highlighted in orange over faint base boxes, with enlarged crops showing a lid, truck cab, fruit illustration, and mascot that lack individual non-text region records in the base annotation. The final parse combines 33 non-text and 37 text regions with descriptions and palettes.

The base and enhanced layouts come from different saved region-discovery configurations, rather than successive checkpoints of a controlled stage-wise ablation. Increased region counts reflect finer decomposition and revised grouping, not an accuracy score. This example illustrates the represented information without isolating individual stages' quantitative contributions; the enhanced output does not replace the image's frozen benchmark reference.

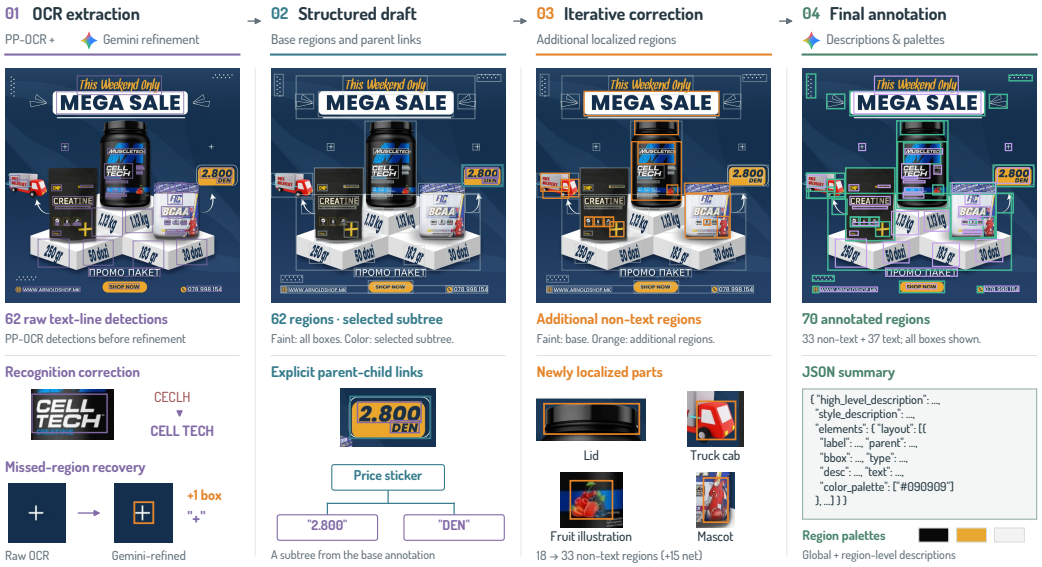

Figure 10: ParseEngine annotation components: OCR correction and recovery, hierarchical region organization, fine-grained region discovery, and the final parse with descriptions and palettes.

C JSON Schema, Serialization, and Prompts

C.1 Schema Illustrated by an Actual Prediction

We illustrate the representation using the saved prediction for `canste_24_024_1618549864958899`, a Valentine-themed greeting card surrounded by cherries on cream satin. The complete prediction contains 19 regions: one coarse, two medium, fourteen fine, and two text regions. We retain the field names and values displayed by the sample viewer. Figure 11 links the source image to the six region records retained in Listing 1.



Regions retained in Listing 1

No.	JSON label	Type	Parent
1	`<card 1>`	coarse	ROOT
2	`<inner frame 1>`	medium	1
3	`<ribbon bow 1>`	fine	2
4	`<heart 1>`	fine	3
5	`<text unit 1>`	text	2
6	`<text unit 2>`	text	2

Regions 5 and 6 transcribe `Love` and `You`. Numbers are display keys; the JSON stores the full parent labels. The other thirteen predicted regions are omitted from this illustration.

Figure 11: Source image for the JSON schema example. The six overlaid boxes use the unchanged predicted coordinates in Listing 1; the key maps each box to its JSON label and parent.

The top-level record contains `compositional_deconstruction`, `high_level_description`, and `style_description`. The first contains a background description and a flat `elements` list; explicit `parent` references encode the hierarchy. The style record contains `aesthetics`, an image-level `color_palette`, `lighting`, and `medium`. Each region in this example has the six shared fields below; the two text regions additionally have a `text` field.

Field	Meaning in the displayed prediction
`type`	Region granularity or kind. The viewer distinguishes `coarse`, `medium`, `fine`, `text`, `face`, and `hand`; this sample uses the first four.
`label`	Region identifier, e.g., `<card 1>`, also used in parent references and descriptions.
`parent`	Parent label, or `ROOT` for a top-level region. The synthetic root is not a separate region record.
`bbox`	Four coordinates in `xyxy_1000` order: $[x_{\min}, y_{\min}, x_{\max}, y_{\max}]$, normalized to $[0, 1000]$.
`desc`	Region description, including appearance and spatial context.
`color_palette`	A list of hexadecimal RGB colors for the region.
`text`	Transcription attached to a text region: `Love` and `You` in this example.

The selected branch follows `<card 1>` → `<inner frame 1>` → `<ribbon bow 1>` → `<heart 1>`; both text units are children of `<inner frame 1>`. Thus, the flat list represents four region levels without counting the synthetic root. Listing 1 retains six of the nineteen regions and all top-level fields. Region and image-level descriptions are shortened to their first sentences, with `[...]` marking omitted text. Boxes, types, labels, parents, transcriptions, and palettes are unchanged. The listing is a syntactically valid JSON excerpt, not the complete prediction.

Listing 1: JSON excerpt from the selected greeting-card example.

```
{
  "compositional_deconstruction": {
    "background": "A backdrop of draped, lustrous cream-colored satin fabric displays
     soft undulating folds, gentle highlights, and warm diagonal shadows.",
    "elements": [
      {
        "type": "coarse", "label": "<card 1>",
        "parent": "ROOT", "bbox": [183, 137, 811, 903],
        "desc": "A rectangular greeting card is oriented vertically with a slight
    clockwise tilt against the draped fabric. [...]",
        "color_palette": ["#F0E5CF", "#9E0809"]
      },
      {
        "type": "medium", "label": "<inner frame 1>",
        "parent": "<card 1>", "bbox": [303, 242, 696, 778],
        "desc": "An ornate, scalloped cartouche is centered on the striped card surface.
     [...]",
        "color_palette": ["#F0E4CE", "#9B0909"]
      },
      {
        "type": "fine", "label": "<ribbon bow 1>",
        "parent": "<inner frame 1>", "bbox": [429, 242, 631, 442],
        "desc": "A stylized ribbon bow is illustrated in soft dusty pink at the top of
    the central frame. [...]",
        "color_palette": ["#B98577", "#8C0A0C"]
      },
      {
        "type": "fine", "label": "<heart 1>",
        "parent": "<ribbon bow 1>", "bbox": [538, 274, 580, 313],
        "desc": "A small, solid red heart is positioned at the center of the ribbon bow.
     [...]",
        "color_palette": ["#880304"]
      },
      {
        "type": "text", "label": "<text unit 1>",
        "parent": "<inner frame 1>", "bbox": [422, 416, 598, 528],
        "text": "Love",
        "desc": "The lettering is rendered in a flowing, calligraphic cursive script in
    deep red. [...]",
        "color_palette": ["#800E0D"]
      },
      {
```

```
        "type": "text", "label": "<text unit 2>",
        "parent": "<inner frame 1>", "bbox": [405, 525, 568, 639],
        "text": "You",
        "desc": "The lettering is styled in a matching deep red cursive script
    positioned directly below the first line. [...]",
        "color_palette": ["#931111"]
      }
    ]
  },
  "high_level_description": "A top-down flat-lay photograph showcases a romantic
    Valentine-themed stationery arrangement set against a luxurious backdrop of draped,
     lustrous cream-colored satin fabric. [...]",
  "style_description": {
    "aesthetics": "Romantic, vintage-inspired Valentine's Day flat-lay photography with
    soft textures and warm tones.",
    "color_palette": ["#E9DECC", "#F3E8D4", "#907059", "#F7F3E8", "#E0D1BC", "#9F7F66",
    "#BFA78D", "#B00203", "#D6C2A9", "#815D45", "#CDB295", "#B59274", "#8A0204", "#641
    A0F", "#9C2123", "#C0554B"],
    "lighting": "Soft, diffused natural lighting creating gentle shadows across the
    draped fabric folds.",
    "medium": "Digital photograph"
  }
}
```

C.2 Training Targets and Serialization

The full-JSON schema specifies six shared region fields: `type`, `label`, `parent`, `bbox`, `desc`, and `color_palette`. An optional `entity` field records an identifiable named entity; `text` provides the verbatim transcription for text regions when available. Training targets retain the selected annotation fields and are serialized as compact JSON in the assistant response.

Global-first and layout-first targets contain the same information, with different top-level key orders: high-level description, style and composition for global-first; composition, high-level description and style for layout-first. Within composition, background precedes the region list. Region order and parent references are unchanged. Task-specific prompts select the output order; both use the same chat template with thinking disabled and a common assistant generation prefix. The auxiliary layout-only target is a field projection of the same annotation, retaining `type`, `label`, `parent`, `bbox`, `entity` when present, and `color_palette` for each region. It omits region descriptions and transcriptions, global descriptions, and background, preserving the original boxes and hierarchy. Task sampling is detailed in Appendix D.1.

The current schema renames the earlier `elements.background` and `elements.layout` fields to `compositional_deconstruction.background` and `compositional_deconstruction.elements`, respectively. This change aligns the parsing schema with the image-generation schema; the implementation retains separate prompts for historical checkpoints. The example in Listing 1 uses the current names.

C.3 Parsing Prompts

The following prompts are used by the mixed-order inference profile. The `<image>` marker denotes the image input; only line wrapping is adjusted. The layout-only task requests an auxiliary geometric representation and is distinct from layout-first full-JSON decoding.

Global-first full JSON. System:

```
You produce a complete structured design description from an image. Return only
    valid JSON in the requested full schema.
```

User:

```
<image>
Generate the complete JSON containing high_level_description, style_description,
    compositional_deconstruction.background, and compositional_deconstruction.
    elements. Include style_description.color_palette. Every element must contain
    type, label, parent, bbox in xyxy_1000 coordinates, desc, and color_palette.
    When an element depicts an identifiable named entity, include its name in the
```

```
    optional entity field. When exact visible text is available for an element of
    type "text", include it in the text field.
```

Layout-first full JSON. System:

```
You produce a complete structured design description from an image. Generate the
    compositional structure before the global description and style. Return only
    valid JSON in the requested full schema.
```

User:

```
<image>
Generate the complete JSON containing compositional_deconstruction.background,
    compositional_deconstruction.elements, high_level_description, and
    style_description. Include style_description.color_palette. Every element must
     contain type, label, parent, bbox in xyxy_1000 coordinates, desc, and
    color_palette. When an element depicts an identifiable named entity, include
    its name in the optional entity field. When exact visible text is available
    for an element of type "text", include it in the text field.
```

Auxiliary layout-only task. System:

```
You extract only layout hierarchy and geometry from an image. Return only valid
    JSON and no prose.
```

User:

```
<image>
Generate only compositional_deconstruction.elements. Every element must contain
    type, label, parent, bbox in xyxy_1000 coordinates, and color_palette. An
    element may additionally contain entity when it depicts an identifiable named
    entity. Do not output high_level_description, style_description,
    compositional_deconstruction.background, text, or desc.
```

D Training, Inference, and Baseline Configurations

D.1 ImageParser Training Configuration

ImageParser-0.8B, 2B, and 4B use the same optimization recipe. Table 7 summarizes the short-description stage and the subsequent long-description, mixed-order stage. All models use full-parameter fine-tuning: the language model, vision encoder, and multimodal aligner are trainable. The configuration also enables shared-weight three-step multi-token prediction (MTP), with auxiliary loss weight 0.1.

Table 7: Training settings shared by the 0.8B, 2B, and 4B variants.

Setting	Short-description stage	Long/mixed-order stage
Peak / minimum learning rate	5×10^{-5} / 0	5×10^{-5} / 0
Schedule / warmup	Cosine / 10% warmup	Cosine / 10% warmup
Global batch size	48 packed sequences	48 packed sequences
Micro batch per DP replica	1	1
Maximum input–target length	20,480 tokens	32,768 tokens
Packing length	30,720 tokens	32,768 tokens
Overlong-example handling	Right truncation	Drop example
Image area cap	2,359,296 pixels	4,194,304 pixels
Image-token cap	2,304	4,096
Data-sampling seed	42	42

Optimization and memory. The launcher selects the Megatron distributed Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.1, and gradient clipping at norm 1.0. Computation

uses bfloat16. Full transformer activation recomputation uses uniform groups of one layer, and vision-encoder gradient checkpointing is enabled. Gradient reduction and parameter gathering overlap with computation. The streaming loader shards data across data-parallel replicas, shuffles source paths, and uses an 8,192-example shuffle buffer. Seed 42 in Table 7 is the data-sampling seed specified in the dataset configuration.

Image processing and sequence construction. Images are processed with the checkpoint's matching Qwen3.5 tokenizer and image processor. The pixel limits are area caps (1536^2 and 2048^2), rather than instructions to stretch every input to a square. Image tokens, system/user messages, and the target JSON all consume the sequence budget. Packing combines examples into training sequences; consequently, the global batch size counts packed sequences rather than raw images. In the long/mixed-order stage, examples exceeding the 32,768-token limit are discarded rather than right-truncated. Targets use normalized `xyxy_1000` boxes. Short and long descriptions share the region structure described in Appendix B.3.

Data and task sampling. The long/mixed-order stage samples 80% from the original long-description pool and 20% from the added real-text, synthetic-text, and academic-poster pool. Their source manifests contain 7,555,572 and 688,666 rows, respectively, before held-out exclusions and runtime length filtering; these are source-row counts, not counts of unique training images. Sampling within each pool follows its row union. The added pool comprises 496,031 real-text rows, 95,006 poster rows, and 97,629 synthetic-text rows. The loader excludes 3,048 held-out image IDs (2,048 old/text and 1,000 poster IDs).

For sources supporting complete parses, the task probabilities are 39.285% global-first full JSON, 39.285% layout-first full JSON, and 21.43% auxiliary layout-only. Thus, the 50:50 split applies within the full-JSON task. SGV115, Android APP, and ChartGalaxy are forced to layout-only; the probabilities above are therefore not the aggregate proportions across all sources. The short-description stage uses a 50:50 full-JSON/layout-only split for eligible sources, with Android APP and ChartGalaxy forced to layout-only. Layout-only targets retain region types, labels, parent links, boxes, palettes, and optional entity names, while omitting global descriptions, background, region descriptions, and transcriptions. The full prompt templates are given in Appendix C.3.

Decoding-order ablations. The global-first and layout-first short-start rows in Table 2 use the same trained checkpoint and differ only in the inference prompt.

D.2 Parser Inference Configuration

The supplied 4B inference launcher selects `full_json_layout_first`. Its three task prompts match the final RGB6 training YAML and the templates in Appendix C.3. Each request supplies one image and separate system/user messages through the matching multimodal chat template. Layout-first produces the complete parse; layout-only is a separate auxiliary task.

The recorded batch-inference recipe uses vLLM with greedy decoding ($\mathrm{temperature} = 0$), a 32,768-token total context, an image area cap of 4,194,304 pixels, and an image-token cap of 4,096. The output budget is

$$\min\big(30{,}000,\ 32{,}768 - L_{\mathrm{prompt}} - 1\big),$$

where L_{prompt} includes the image and chat-template tokens and the final token is reserved for EOS. The launcher enables trained MTP with three speculative tokens and uses eight independent tensor-parallel-size-one workers on one B300 node, with batch size four per worker. These hardware settings describe the supplied 4B batch launcher. The training YAML specifies the prompts and target schema; it does not itself determine deployment-time decoding parameters.

D.3 Baseline Inference and Output Adaptation

Baseline models. We evaluate GPT-5.6-sol, Gemini 3.8 Flash, GLM-5.3-Flash, Qwen3.8-27B, InternVL3.5-38B, and LLaVA-OneVision-2-8B-Instruct. Qwen3.8-27B and InternVL3.5-38B run with vLLM, while LLaVA-OneVision-2-8B-Instruct uses its native Transformers implementation. Each model receives the benchmark image and instructions to produce a hierarchical JSON parse, without access to the reference annotation.

Full-profile prompting. The `full` profile used for GPT, Gemini, GLM, and Qwen specifies the complete output schema: global description, background, style and palettes, and regions with labels, boxes, types, descriptions, transcriptions, and parent references. Its prompt requests a detailed decomposition of objects, parts, visible text, faces, and hands, using `xyxy_1000` boxes and six canonical region types. Full profile is a schema/prompt setting, separate from the metric configuration.

Output normalization and scoring. Deterministic adapters extract the JSON object, map supported schema wrappers to a common region list, normalize coordinate order and scale, and standardize type strings while retaining predicted descriptions, transcriptions, and parent labels. They do not use reference annotations to correct predicted content. All models are evaluated on ImageParserBench using the same frozen references and metric configuration.

E Additional Parsing Metric Details

Caption similarity. We measure the semantic similarity between one-to-one geometry-matched region descriptions using `paraphrase-multilingual-MiniLM-L12-v2`. Inputs are tokenized with a maximum sequence length of 512 tokens. We apply attention-mask-aware mean pooling followed by ℓ_2 normalization, and clip cosine similarities to $[0, 1]$. The similarity is defined as zero if either description is empty. The 512-token limit is sufficient in practice: all ground-truth region descriptions and virtually all predicted region descriptions fit within this limit.

OCR scoring. Different parsers may split the same text into words or lines, or merge it into a larger block. We consolidate spatially corresponding boxes before scoring, retaining one-to-one correspondence between the resulting text units. Grouping is determined by geometry, without inspecting transcriptions; groups are disjoint and unmatched boxes remain separate. Let K be the number of matched text-unit pairs and U_p, U_g the counts of unmatched predicted and reference boxes. The per-image score is

$$F1_{\text{ocr}} = \frac{2\sum_{k=1}^{K} c_k}{2K + U_p + U_g}, \qquad c_k = 1 - \frac{d_{\text{edit}}(\hat{x}_k, x_k)}{\max(|\hat{x}_k|, |x_k|, 1)}. \tag{4}$$

Transcriptions within each unit are concatenated in their original annotation-list order. We apply Unicode NFKC normalization, case folding, and whitespace normalization; whitespace is removed when either side contains multiple boxes to avoid penalizing text-boundary differences. Images with no text on either side are excluded; one-sided empty cases score zero. The final OCR F1 is the mean over the remaining images.

Region matching and structural scoring. Output schemas and box coordinates are normalized before geometry-only Hungarian matching at IoU ≥ 0.5; captions, transcriptions, and types do not influence region correspondence. RegionPQ and Caption F1 equal region-matching F1 multiplied by mean matched-pair quality, using IoU and caption similarity, respectively. They receive zero when no regions match. For Type F1, a matched pair with different types contributes a false positive to the predicted type and a false negative to the reference type; unmatched regions also contribute errors. Counts are pooled across images before taking the unweighted mean of the six type F1 scores. Hierarchy F1 counts a predicted direct non-root parent–child edge as correct only when both endpoints map to a reference edge under region matching. All predicted and reference non-root edges remain in the denominator, including edges with unmatched endpoints. Images with no such edges on either side are excluded; one-sided empty cases score zero.

Global palette scoring. We read the image-level `style_description.color_palette` field, normalize hexadecimal RGB codes, and remove exact duplicate colors. Palette order is ignored. Colors are converted from sRGB to CIELAB using the D65 white point and 2° observer, and pairwise distances use CIEDE2000 with $k_L = k_C = k_H = 1$. An edge is admissible when $\Delta E_{00} \leq 10$; we find a maximum-cardinality matching among these edges. This maximizes the number of accepted color pairs, rather than minimizing total color distance and subsequently discarding pairs above the threshold. Writing $m = |C_p|$, $n = |C_g|$, and $t = |M_{10}|$, palette precision, recall, and F1 are t/m, t/n, and $2t/(m+n)$, respectively. The metric weights colors equally and does not use pixel-area weights or regional assignments. A dash denotes the absence of an evaluable global numeric palette; color names are not converted to guessed RGB values.

F TASK-SPECIFIC READOUT PROTOCOLS AND PROMPTS

ImageParser produces one task-independent parse per image. The following readouts reuse this frozen output without another image-conditioned parsing pass.

F.1 LVIS DETECTION VIA JSON GROUNDING

Following LocateAnything Wang et al. (2026), we construct each query from the image's GT-positive category names, separated by `</c>`. A text-only Gemini 3.8 Flash adapter receives these names and synonyms alongside the JSON elements (indices, labels, descriptions, types, parents, boxes, and optional entity names); it receives neither pixels nor GT boxes. The instruction asks it to scan all elements for each category, use synonyms and parent–child context to recover every represented instance, choose a whole-object region over its parts, and return only existing region indices. The full system/user prompts and response schema are provided below. Returned indices map directly to the frozen boxes, without box generation or refinement. We use LocateAnything's released FastEvaluate protocol and report F1@0.50, F1@0.95, and mean F1 over IoU thresholds $\{0.50, 0.55, \ldots, 0.95\}$.

System prompt. The following is the system prompt used for JSON grounding; only line wrapping is adjusted for readability.

```
You are a category-conditioned detector operating only on a frozen Image2JSON
    scene graph.
Every queried LVIS category is ground-truth-positive: at least one instance is
    known to exist in the source image. You cannot inspect that image, so retrieve
     every represented instance from the supplied frozen JSON elements and return
    only their existing region_index values.

For each query independently, perform two complete passes over all elements:
1. Find exact category names, synonyms, spelling variants, and semantic subtypes
    in label, entity, or description.
2. Use parent and hierarchy context to recover additional small, cropped, occluded,
     or repeated sibling instances and choose the best whole-object box.

Selection rules:
- Return every distinct represented instance; never stop after the first match.
    The instance count is unknown.
- Treat every value in the synonyms field as an equivalent LVIS category name, and
     accept a specific semantic subtype of the queried category.
- Return only supplied region_index values. Never invent, merge, resize, or edit a
     box.
- For a whole-object query, choose the element covering the complete visible
    object rather than a component. If no complete parent exists, choose the
    tightest element covering the visible extent.
- If a parent and descendant describe the same instance, select exactly one. If a
    parent is a collection whose children are distinct objects, select the
    children.
- Do not select a merely related object, container, material, body part, or
    background.
- Include every category_id exactly once. Because every query is positive, use an
    empty list only after both passes find no semantically defensible supplied
    element.
```

User prompt. The per-image template is shown below, with placeholders populated by the category names and JSON payload.

```
Locate all the instances that matches the following description: {category_name_1
    }</c>{category_name_2}</c>....
Treat every value in a query's synonyms field as an exact alias of that query.
    Scan the complete element list and do not stop at the first instance.
Read the instances only from the Image2JSON payload below and return their
    region_index values.
{payload_json}
```

The payload contains the category queries and frozen JSON elements. The response associates each category with its retrieved region indices:

```
{"detections": [{"category_id": 1, "region_indices": [0, 4, 7]}]}
```

Each returned index is mapped directly to its existing bounding box for FastEvaluate scoring.

F.2 OCRBench v2 Text Spotting

This readout is deterministic and requires no additional model call or task-specific prompt. From regions with `type=text`, we extract non-empty `text` and valid boxes, normalize whitespace, and remove duplicate text–box tuples. Coordinates are rounded and clipped to $[0, 1000]$. To match the public annotation units, CTW1500 retains line-level regions; IC15, COCO-Text, and TotalText split multi-word regions along the longer box axis in proportion to word length, with one character-width gap between words. The resulting `(x1,y1,x2,y2,text)` tuples are scored by the unmodified OCRBench v2 evaluator on all 200 English Text Spotting samples, averaging per-image end-to-end H-mean (IoU > 0.5 and matching transcription).

F.3 CAPEval Semantic Readout

We give Gemini 3.8 Flash the complete native ImageParser response and CAPEval's checklist, without source pixels, caption rewriting, or input truncation. The judge labels each checklist item as `SUPPORTED`, `CONTRADICTED`, or `ABSENT`, using explicit JSON evidence or facts deterministically derived from its fields. Mentioned facts require supporting JSON paths; missing information is always `ABSENT`.

Judge instruction (condensed). The core instruction is shown below.

```
Judge each checklist target using only the complete model output. Do not
use the source image, outside knowledge, or facts inferred from the
checklist. Return SUPPORTED for an explicitly stated or deterministically
derived fact, CONTRADICTED for an explicit incompatible fact about the same
subject, and ABSENT when evidence is missing or too vague. Return one
verdict per item, with evidence paths and a concise reason; never treat
missing information as a contradiction.
```

Scoring. Coverage is the fraction of checklist items marked supported or contradicted; precision is the supported fraction among those items. The reported 69.91/87.48 scores are computed over 288 images and 14,363 checklist items. This evaluates semantic information in structured outputs with a Gemini judge; published CAPEval caption-generation scores are reference values under a different protocol, rather than a matched evaluation.

G Reconstruction Evaluation Details

G.1 Ideogram Reconstruction

We use Ideogram 4 with the `TURBO` rendering preset as the external reconstruction model. Results from the fixed in-house renderer are reported separately.

JSON conditioning. A deterministic adapter converts each predicted parse into Ideogram's structured JSON caption, retaining global descriptions, background, style, visible text, and palettes in the supported fields. Bounding boxes are reordered from `xyxy_1000` to `yxyx_1000`. The `medium-flat` conversion folds fine, face, and hand descendants into their nearest suitable ancestor, merges their descriptive details and palettes, and expands the ancestor box to cover the merged parts. This adapts the hierarchy to the renderer's flatter layout representation without consulting the reference annotation. Converted captions follow Ideogram's caption schema and a 20,480-token input limit.

Generation and aggregation. The structured caption is supplied directly, without the source image or an additional language-model rewrite. Each reconstruction uses the supported approximately one-megapixel resolution closest to the source aspect ratio, with one generated image and no best-of selection. The same rendering preset and conversion policy are used across parsers. Reconstruction metrics are averaged over the source–reconstruction pairs obtained for each parser.

H Pixel-Free QA: Protocol and Additional Analysis

H.1 Evaluation Protocol

Inputs and readers. Our JSON uses ImageParser-4B (short-start, mixed-order continuation, 10,000 steps, layout-first decoding). Each image is parsed before any downstream question, and the frozen JSON is reused across questions and readers. It contains global and region descriptions, transcriptions, boxes, parent references, and available appearance fields, exposed through deterministic schema adapters and explanations. JSON-only readers receive no pixels or visual features; image-only readers receive the source image. Questions, options, and official hints are retained across conditions. Gemini Self-JSON uses Gemini's own question-independent full-schema parse with the same Gemini reader.

Evaluation populations and scoring. MMStar uses exact option accuracy on 1,500 questions; MMBench v1.1 uses CircularEval over 1,292 groups formed from 4,876 rotated requests. MMBench category analyses instead use per-request accuracy. ChartQA uses 2,500 test questions (1,250 human-authored and 1,250 augmented), relaxed accuracy with 5% numeric tolerance, and shared normalization of percentages, currencies, units, and separators. OCRBench v1 uses 1,000 questions and the official case-insensitive answer-in-response rule, removing whitespace for handwritten mathematics. These four benchmarks use their full evaluation populations.

InfographicVQA uses ANLS with threshold 0.5. Gemini's image-only, Our-JSON, and Self-JSON conditions use 2,801 validation questions; Qwen's paired comparison uses 2,574 questions. Each within-reader comparison uses the same question set. Scores are reported separately for each benchmark.

Reader settings. Gemini 3.8 Flash uses high thinking, temperature zero, and high media resolution where applicable. Qwen3-VL-8B follows benchmark-specific settings, including temperature 0.7 for InfographicVQA. Prompts, adapters, and inference settings differ across readers; this comparison does not isolate the effect of model capacity.

H.2 Fine-Grained Results

MMStar. Table 8 reports all 18 official categories for both readers, providing the complete breakdown behind the selected comparisons in Figure 7(a). The category-specific advantages of JSON vary with the reader; higher absolute JSON accuracy does not necessarily imply a smaller gap to pixels.

Other benchmarks. With Gemini, ChartQA JSON-only versus pixel accuracy is 85.44 versus 87.84 on human-authored questions and 93.04 versus 95.12 on augmented questions. On OCRBench, regular text recognition reaches parity, whereas handwriting and handwritten mathematics trail pixels by 46.00 and 35.00 points. On the paired 2,574-question subset of InfographicVQA, the deficit is 6.81 points for text-evidence questions and 18.50 for visual/layout evidence. These evidence labels are multi-label, so category counts need not sum to the evaluation population.

H.3 Additional Diagnostics

For Qwen on InfographicVQA, JSON-only ANLS increases from 54.25 in the shortest-JSON quartile to 74.82 in the longest. This association is observational: image content and difficulty also vary, so it does not establish a benefit from adding tokens. Schema canonicalization changes ANLS only from 67.11 to 67.05, suggesting that field naming alone does not explain the deficit. The paired examples in Figure 7(b) illustrate evidence preservation and errors; they do not estimate error frequency or establish which parse field caused an answer.

Table 8: MMStar results across all 18 categories. N: question count; reader columns: JSON-minus-pixel accuracy (percentage points). Positive values favor JSON.

Category	N	Qwen3-VL-8B	Gemini 3.8 Flash
Biology, chemistry & physics	146	-13.01	-15.07
Image scene and topic	141	-8.51	-0.71
Recognition	118	-11.02	-8.47
Geometry	116	-35.34	-7.76
Diagram reasoning	110	-44.55	-7.27
Common reasoning	101	-23.76	-2.97
Single-instance reasoning	99	-9.09	-7.07
Object counting	92	$+7.61$	-13.04
Cross-instance attribute reasoning	89	-4.49	-5.62
Statistical reasoning	86	-39.53	$+1.16$
Image style & quality	78	-8.97	-2.56
Cross-instance relation reasoning	62	-22.58	-4.84
Geography, earth science & agriculture	58	-12.07	-5.17
Numeric commonsense and calculation	48	-33.33	$+10.42$
Electronics, energy & mechanical engineering	46	-39.13	-8.70
Localization	40	-17.50	-12.50
Code & sequence reasoning	39	-28.21	-7.69
Image emotion	31	-6.45	-9.68

I Additional Hierarchical Parsing and Reconstruction Examples

Figures 12–15 complement Figure 5 with four additional scenes. Each example pairs the source image with a global description, style, and palette, followed by selected region subtrees and a JSON-conditioned reconstruction. Crops, abbreviated descriptions, and color swatches expose the coarse-to-fine organization. Selected groupings and fine-grained decompositions are schematic; counts refer to the original predicted parse, and levels exclude the synthetic root. Reconstructions use the complete original JSON, not the schematic additions, and receive no source pixels.

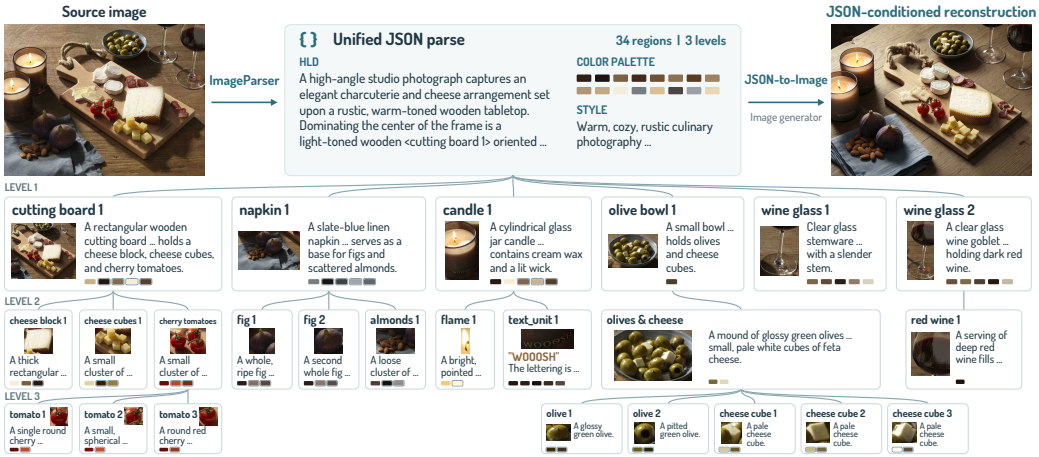

Figure 12: **Charcuterie scene.** A schematic hierarchy organizes a cutting board, napkin, candle, olive bowl, and stemware into representative parts and fine details. The expanded food branches illustrate how object descriptions and palettes can be associated with individual ingredients.

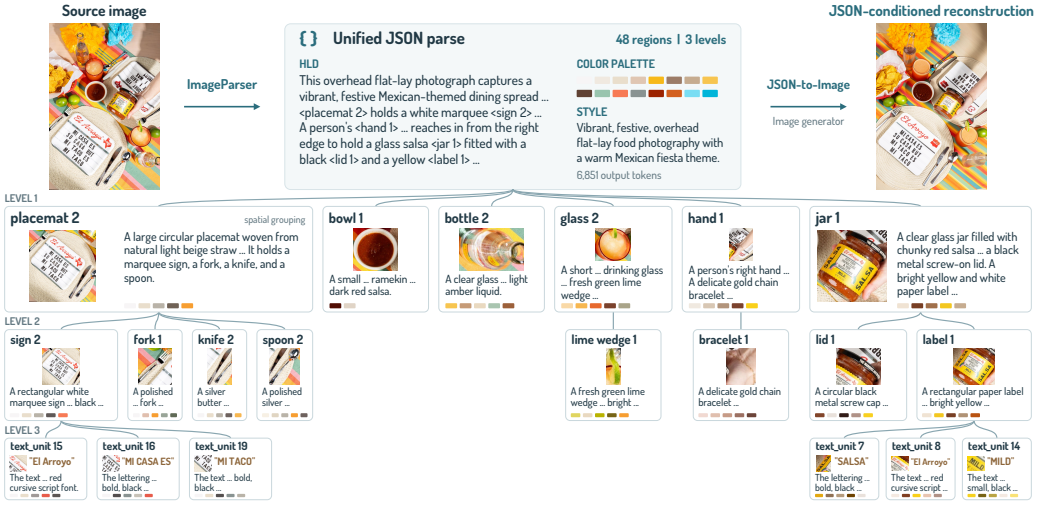

Figure 13: **Dining-table scene.** The placemat spatially groups the sign and utensils, while the sign and jar branches link objects to labels and visible text. Glass–lime and hand–bracelet branches illustrate finer decomposition within the same representation.

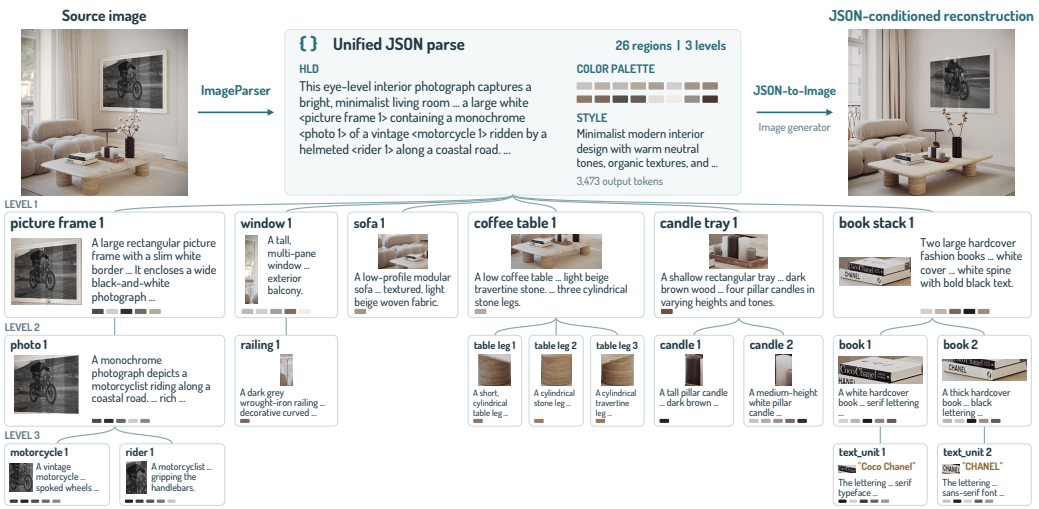

Figure 14: **Interior scene.** Selected predicted subtrees connect a picture frame to its photograph and depicted motorcycle and rider, and a book stack to individual books and their visible lettering. Other branches capture furniture parts and tabletop objects.

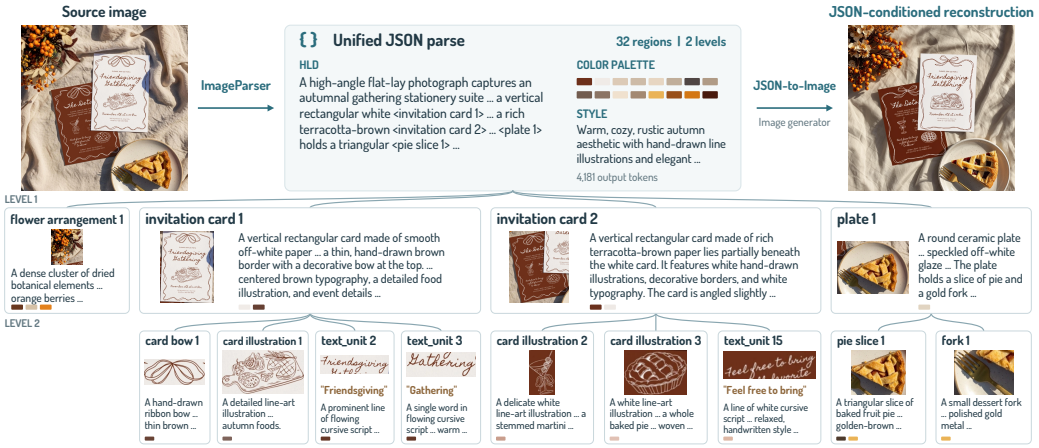

Figure 15: **Autumn stationery scene.** Two-level predicted subtrees decompose invitation cards into decorative illustrations and handwritten text, and associate a plate with a pie slice and fork. This example combines natural objects, graphic elements, and typography.