DiffTable: Benchmarking Grounded Difference Detection in Table Images

Anonymous ACL submission

Abstract

Visual table difference detection is important for auditing and revision tracking, but existing table benchmarks mainly study single-table understanding and visual-difference benchmarks rarely require structured table outputs. We introduce DiffTable, the first benchmark dedicated to paired table image difference detection with structured localization. Given two rendered table images, models must output typed JSON objects that identify value edits, row/column changes, cell-span changes, or no difference, with normalized boxes and edited texts when applicable. DiffTable is built from controlled edits to public table sources followed by browser rendering, yielding rendering-derived boxes; source-table leakage checks confirm zero overlap between training and evaluation splits. We evaluate proprietary and open-source VLMs, OCR-to-LLM and oracle-HTML pipelines, and fine-tuned 7B VLMs using an end-to-end metric that requires correct type, localization, and text recovery. Zero-shot and OCR-based systems often recognize the task but fail strict localization; rotation, blur, cropping, and compression further challenge table-structure and cell-bbox recovery. A two-stage fine-tuned Qwen2.5-VL-7B model with OCR-style warmup reaches 0.693 E2E-F1 on ID-Hard-300, 0.890 on ID-Full, and 0.605 on OOD-300.

1 Introduction

Visual table difference detection asks a system to compare two rendered versions of a table and identify what changed and where. Tables are widely used in financial reports, scientific papers, administrative documents, and web pages, where even small revisions can affect downstream decisions: a revenue entry may change from $100M to $150M, or a merged header may shift which columns share a fiscal period. Reliable table comparison is therefore important for auditing, verification, and revision tracking.

The difficulty is not merely deciding whether two table images differ. In auditing workflows, a response such as "the table has changed" or a free-form caption such as "some values are different" is hard to verify. The useful output is structured evidence: the affected cell or row, its location in each image, the original and modified text when applicable, and a type label distinguishing value tampering from structural edits. This makes table difference detection a grounded comparison problem rather than only image-level classification or captioning.

Existing table understanding benchmarks mainly focus on single-table tasks. PubTabNet and FinTabNet support image-based table structure recognition from scientific and financial documents (Zhong et al., 2020; Zheng et al., 2021); WikiTableQuestions, HiTab, TAT-QA, and FeTaQA evaluate different forms of reasoning over structured or semi-structured tables (Pasupat and Liang, 2015; Cheng et al., 2022; Zhu et al., 2021; Nan et al., 2022); and recent table-oriented LLM or multimodal benchmarks study complex table QA and multimodal instruction following, including over table images (Wu et al., 2025; Zheng et al., 2024). These settings have advanced table parsing and reasoning, but they do not directly test whether a model can compare two rendered table images and produce a structured record of their differences.

Pairwise visual comparison has also been studied in document image change detection and image difference captioning. VisualDiff and later document-image work aim to detect changed regions between document versions (Jain and Doermann, 2013; Pun et al., 2023), while recent OCR-free text change detection compares multilingual document text regions directly in image space (Park et al., 2025). Image difference captioning works and benchmarks ask models to describe differences between paired natural images (Jhamtani and Berg-Kirkpatrick, 2018; Di et al., 2025; Hu et al., 2024;

Liu et al., 2025). However, table difference detection introduces different output requirements. For downstream auditing systems, a natural-language description or coarse heatmap is insufficient: the model should output typed, localized, and machine-readable difference objects over table cells, rows, columns, and cell spans.

DiffTable targets this gap as the first benchmark dedicated to visual table difference detection with structured localization. Given two rendered table images, a model is required to output a JSON-formatted list of difference objects. Each object specifies the difference type, the affected bounding box on Table A and/or Table B, and, for value edits, the original and modified texts. DiffTable covers value tampering, row/column insertion or deletion, cell-span change, and no-difference examples. DiffTable therefore asks models to turn visual comparison into verifiable structured evidence, not merely to notice that a table changed.

Building such a benchmark requires more than collecting paired screenshots. The data must specify which controlled edit operation was applied, provide localized regions for the affected rendered table areas, and use splits that avoid near-duplicate leakage. DiffTable addresses these requirements by applying controlled edits to structured public table sources before browser rendering. We use FinTabNet, PubTabNet, HiTab, and WikiTableQuestions as in-domain sources, and SciTSR, TAT-QA, and FeTaQA as unseen out-of-domain sources (Zheng et al., 2021; Zhong et al., 2020; Cheng et al., 2022; Pasupat and Liang, 2015; Chi et al., 2019; Zhu et al., 2021; Nan et al., 2022). Because edits are applied before rendering, ground-truth bounding boxes can be extracted from rendered table cells rather than annotated manually or inferred by OCR. Source-table leakage checks confirm zero overlap between training and evaluation splits.

We use DiffTable to evaluate zero-shot proprietary and open-source VLMs, OCR-to-LLM pipelines, oracle HTML settings, and fine-tuned 7B VLMs, separating coarse recognition from machine-usable localization and text recovery. The OOD split adds rotation, blur/noise, cropping, and compression, highlighting reliable table-structure and cell-bbox recovery as a bottleneck for OCR-based pipelines.

Our contributions are threefold. First, we introduce the first structured localization benchmark for visual table difference detection over paired rendered table images. Second, we construct DiffTable with controlled edits, rendering-derived bounding boxes, source-table leakage checks confirming zero training-evaluation overlap, and both in-domain and out-of-domain evaluations, making the task scalable and reproducible. Third, we provide a comprehensive evaluation showing that current general-purpose VLMs remain limited under strict structured localization, while task-specific fine-tuning with cell-level warmup substantially improves performance.

2 Related Work

2.1 Table understanding and visual table parsing

Table understanding has been studied through structure recognition, table QA, and table-to-text generation. PubTabNet and FinTabNet support image-based table recognition (Zhong et al., 2020; Zheng et al., 2021), while WikiTableQuestions, HiTab, TAT-QA, and FeTaQA evaluate reasoning over diverse table formats (Pasupat and Liang, 2015; Cheng et al., 2022; Zhu et al., 2021; Nan et al., 2022). Recent table-focused LLM and multimodal benchmarks further stress table reasoning and instruction following (Wu et al., 2025; Zheng et al., 2024). These works mainly focus on a single table, whereas DiffTable requires pairwise comparison and structured localization.

2.2 Document and image difference detection

Document image change detection localizes changed content between document versions (Jain and Doermann, 2013; Pun et al., 2023), and recent work explores OCR-free text change detection in document images (Park et al., 2025). Image difference captioning works and benchmarks describe visual changes in natural language (Jhamtani and Berg-Kirkpatrick, 2018; Di et al., 2025; Hu et al., 2024; Liu et al., 2025). DiffTable shares the paired-image setup, but focuses on rendered tables and requires typed JSON objects with cell-level boxes and edited texts rather than generic regions or captions.

2.3 Structured grounding and OCR-based comparison

DiffTable also relates to multimodal grounding and structured document parsing, including intermediate-box reasoning, structured visual text parsing, and grounded document editing (Shao et al., 2024; Yu et al., 2026; Suri et al., 2024). A

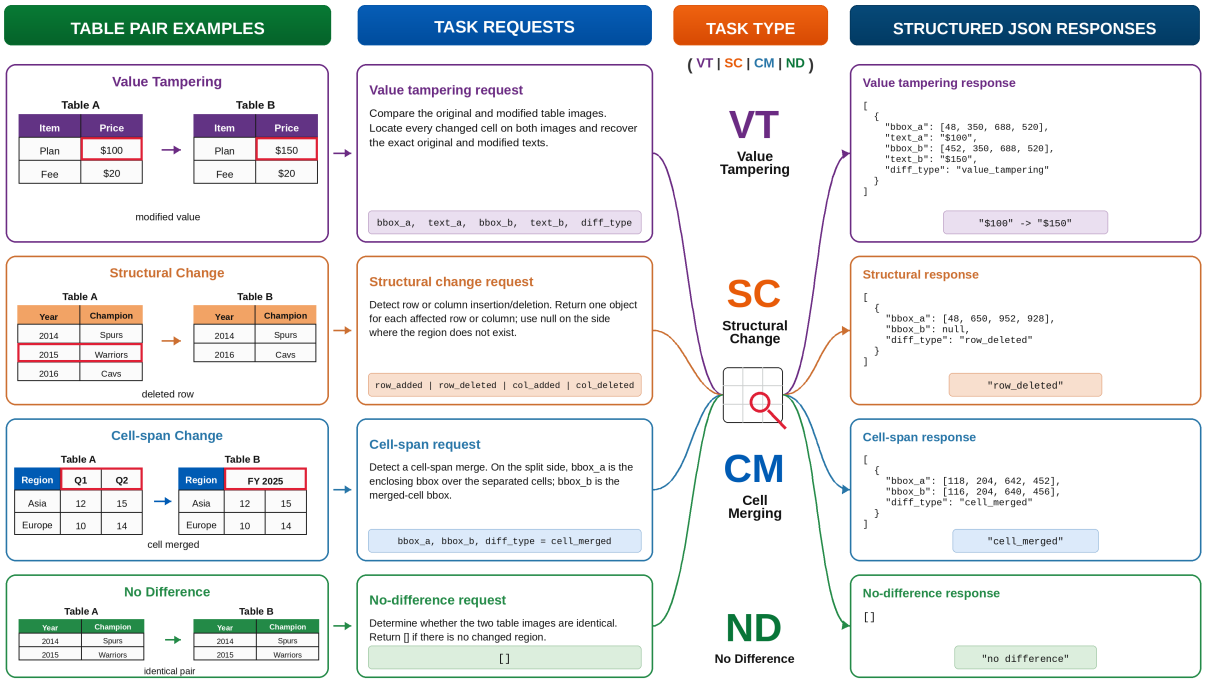

Figure 1: DiffTable task formulation. Given paired rendered table images, models output typed JSON difference objects with table-region boxes and edited text when required.

natural baseline is OCR-to-LLM comparison using extracted HTML or cell structures; recent systems such as MinerU target high-resolution text, layout, and table extraction, while PaddleOCR provides OCR detection and localization components (Niu et al., 2025; Cui et al., 2025). DiffTable evaluates this pipeline alongside oracle HTML and direct image models, aligning with recent calls for table evaluation that accounts for both textual and structural differences (Pancholi et al., 2025).

These prior benchmarks are designed for different goals: single-table parsing and reasoning, document-level change localization, natural-image difference description, or text-region change detection. DiffTable is complementary to these settings. It makes paired rendered-table comparison the primary task and requires the output to be a machine-readable set of typed difference objects grounded to table regions, with edited text recovery when applicable and explicit no-difference cases. This task definition is intended to support audit-style verification rather than to rank prior benchmarks along dimensions they were not designed to optimize.

3 The DiffTable Benchmark

3.1 Task Formulation

The task formulation is motivated by downstream auditing: outputs must be structured while grounding each change in visual evidence. A DiffTable instance contains two rendered table images: Table A is the original table and Table B is the modified table. The model outputs a JSON list of difference objects, each with a fine-grained type and one or two bounding boxes depending on whether the affected region exists in Table A, Table B, or both. Value tampering additionally requires the original and modified texts, while no-difference examples require an empty list.

DiffTable covers four high-level categories: value tampering, structural row/column insertion or deletion, cell-span change, and no difference. Cell-span examples are encoded with the fine-grained type `cell_merged`: Table A contains the split cells and Table B contains the corresponding merged cell. Each example contains one high-level category, although it may contain multiple objects of that category.

Bounding boxes are represented by normalized coordinates in the range $[0, 1000]$, with each box independently normalized to the width and height of its corresponding image. For row or column additions, the affected region exists only in Table B. For row or column deletions, it exists only in Table A. For value tampering and cell-span changes, both sides are expected to have non-null boxes. For `cell_merged`, Table A is the split side and Table B

Split	ID Sources				OOD Sources			Total Samples	Tasks				Input Complexity		Avg. #GT Obj. (+)
	FinTab Net	PubTab Net	HiTab	WTQ	Sci TSR	TAT QA	FeTa QA		VT	SC	CM	ND	Avg. Cells	Avg. Text Chars	
Train	8,000	15,200	5,194	1,927	–	–	–	30,321	10,405	8,802	8,082	3,032	–	–	1.87
Aux. Warmup	2,276	3,396	1,031	576	–	–	–	7,279	–	–	–	–	–	–	–
ID-Full Test	1,000	1,900	629	242	–	–	–	3,771	1,271	1,098	1,010	392	63	489	1.85
ID-Hard-300	78	154	50	18	–	–	–	300	90	90	90	30	**155**	**1,160**	**2.09**
OOD-300	–	–	–	–	100	100	100	300	84	102	84	30	81	725	1.83

Table 1: Dataset composition and input complexity for DiffTable. ID-source columns report samples derived from FinTabNet, PubTabNet, HiTab, and WikiTableQuestions (Zheng et al., 2021; Zhong et al., 2020; Cheng et al., 2022; Pasupat and Liang, 2015); OOD-source columns report samples from SciTSR, TAT-QA, and FeTaQA (Chi et al., 2019; Zhu et al., 2021; Nan et al., 2022). Total Samples counts paired table-image examples, except Aux. Warmup, which contains single-table cell text-and-bbox supervision examples derived from rendered tables rather than OCR outputs. VT, SC, CM, and ND denote value tampering, structural change, cell-span/merged-cell change, and no-difference examples, respectively; these task distributions are computed from ground-truth answer objects. Avg. Cells is the mean number of parsed HTML table cells in the A-side source table, and Avg. Text Chars is the mean total character count of all A-side cell texts; both summarize input complexity for evaluation splits. Avg. #GT Obj. (+) is the mean number of ground-truth answer objects in positive paired-difference examples, excluding no-difference examples whose answer list is empty.

is the merged side; box_a encloses all affected split cells, and box_b is the rendered box of the merged cell.

Figure 1 illustrates the paired-image input, task-specific request, high-level task type, and structured JSON response format used by DiffTable.

3.2 Data Construction

A central challenge in building DiffTable is obtaining exact labels without reducing the task to artificial table grids. Manual annotation would be expensive and noisy, while OCR-derived boxes would entangle the benchmark labels with OCR errors. We therefore construct DiffTable from structured public table sources: FinTabNet, PubTabNet, HiTab, and WikiTableQuestions (Zheng et al., 2021; Zhong et al., 2020; Cheng et al., 2022; Pasupat and Liang, 2015). We apply controlled edits before rendering and then obtain labels from the rendered DOM. This preserves realistic rendered-table appearance while allowing the generator to record the applied edit type and the rendered region affected by that edit, reducing annotation noise as a confounding factor in evaluation.

Figure 2 shows the construction pipeline: we parse source tables, filter invalid or overly complex cases, apply one supported edit operation, render the original and modified images, and extract ground-truth boxes from DOM positions.

Edit generation is controlled but content-preserving where possible. Value tampering samples visible non-empty cells and perturbs digits or alphabetic characters. Structural insertions are generated by reversing row or column deletion, so inserted regions contain natural table content rather than synthetic filler. Cell-span examples merge adjacent non-header cells and use rendering-derived union boxes for the Table-A split side. Details are in Appendix A.

We run source-table leakage checks confirming zero overlap between training and evaluation splits, along with integrity checks for missing images, parsing failures, and invalid boxes.

3.3 Dataset Splits

DiffTable contains a main training set, an auxiliary warmup set, ID-Full, ID-Hard-300, and OOD-300. Table 1 summarizes their source composition, task distribution, and input complexity. ID-Hard-300 is a stratified stress-test subset emphasizing larger tables, while OOD-300 uses SciTSR, TAT-QA, and FeTaQA (Chi et al., 2019; Zhu et al., 2021; Nan et al., 2022), which are unseen during training.

OOD-300 tests both source transfer and visual-domain robustness. Besides using unseen sources, it applies broader templates, fonts, CSS styles, and image perturbations such as rotation, translation, scaling, cropping, texture, blur/noise, color shifts, and stronger JPEG compression. Boxes and oracle HTML are updated after each transformation.

4 Evaluation Protocol

4.1 Metrics

DiffTable evaluation includes sample-level task recognition and object-level end-to-end matching.

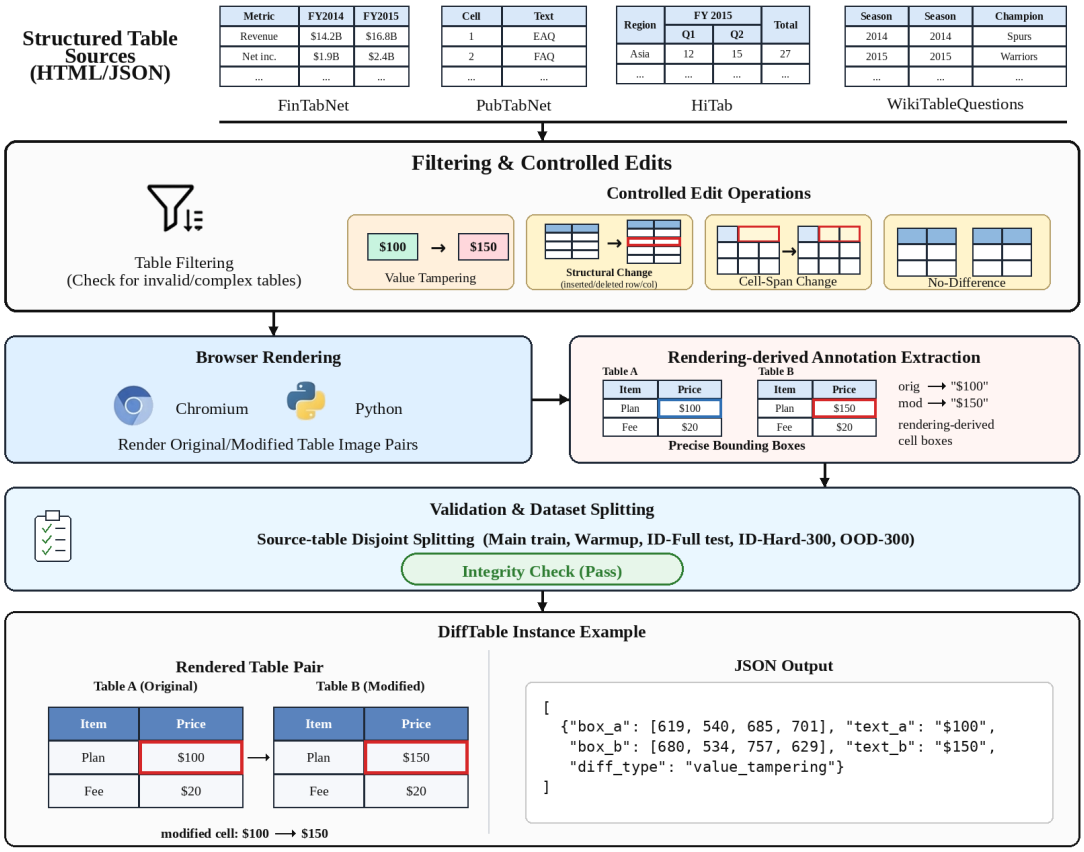

Figure 2: DiffTable construction pipeline. Structured source tables are filtered, edited before rendering, rendered into paired images, and exported with rendering-derived annotations.

Task accuracy maps each prediction to one of four high-level labels. The primary metric is E2E-F1: predictions and ground-truth objects are matched greedily by pair IoU within the same fine-grained type, and a match is counted as a true positive only if it satisfies the type-specific localization requirement. Value tampering additionally requires exact matches of both edited texts. We also report macro-F1, mIoU, IoU@0.5, and Text-EM; full metric definitions are in Appendix C, with numbered equations for the main metrics in Equations C.1–C.11.

This evaluation is intentionally stricter than image-level change classification. A prediction that identifies the right high-level task but points to the wrong cell is not useful for audit, and a well-localized value edit without the correct before/after text still fails to provide a verifiable record. The metrics therefore separate task recognition from localization-sensitive end-to-end performance.

4.2 Models and Settings

We evaluate zero-shot image prompting, OCR-to-LLM pipelines, oracle HTML inputs, and fine-tuned VLMs initialized from Qwen2.5-VL-7B-Instruct (Bai et al., 2025). The OCR setting constructs OCR HTML by combining MinerU-style table serialization with PaddleOCR-derived localization (Niu et al., 2025; Cui et al., 2025). MinerU provides compact table/HTML organization and cell text, while PaddleOCR-derived boxes provide normalized localization because the MinerU API output used here does not expose per-cell boxes. Injecting these boxes into MinerU-format HTML keeps the OCR HTML setting fully automatic, with possible recognition, structure, and localization errors. Oracle HTML uses rendering-derived structured input as a diagnostic setting. DiffTable-7B is trained in two stages: an auxiliary OCR-style warmup stage maps a single table image to rendering-derived visible cell texts and bboxes, and the paired-task stage maps two table images to DiffTable difference JSON. Training details are in Appendix F.

Method	Input	Task Acc	Macro-F1	mIoU	IoU@0.5	Text-EM	E2E-F1	E2E-P	E2E-R
GPT-4o	Image	0.370	0.343	0.050	0.008	0.259	0.002	0.004	0.002
Qwen3.6-Plus	Image	0.640	0.604	0.145	0.047	0.641	0.027	0.028	0.027
Qwen2.5-VL-7B-Instruct	Image	0.183	0.148	0.010	0.000	0.175	0.000	0.000	0.000
OCR + GPT-4o	OCR HTML	0.660	0.638	0.327	0.194	0.780	0.116	0.142	0.098
OCR + Qwen3.6-Plus	OCR HTML	0.743	0.695	0.335	0.223	0.766	0.147	0.145	0.149
Direct FT	Image	**0.980**	0.968	0.602	0.635	0.898	0.588	0.610	0.568
DiffTable-7B	Image	**0.980**	**0.971**	**0.680**	**0.733**	**0.926**	**0.693**	**0.713**	**0.675**
GPT-4o	HTML Oracle	0.927	0.911	0.758	0.714	0.963	0.639	0.691	0.595
Qwen3.6-Plus	HTML Oracle	0.977	0.970	0.826	0.798	0.978	0.766	0.776	0.757

Table 2: Main results on ID-Hard-300, the complexity-biased in-domain subset. Bold numbers mark the best non-oracle results; HTML Oracle is a diagnostic setting with ground-truth structured HTML.

Method	Input	Task Acc	mIoU	E2E-F1
GPT-4o	Image	0.423	0.069	0.003
Qwen3.6-Plus	Image	0.647	0.195	0.043
Qwen2.5-VL-7B-Instruct	Image	0.177	0.001	0.000
OCR + GPT-4o	OCR HTML	0.637	0.255	0.046
OCR + Qwen3.6-Plus	OCR HTML	0.580	0.342	0.079
Direct FT	Image	**0.950**	0.607	0.530
DiffTable-7B	Image	**0.950**	**0.642**	**0.605**
Qwen3.6-Plus	HTML Oracle	0.977	0.789	0.734

Table 3: Out-of-domain results on OOD-300 from unseen table sources: SciTSR, TAT-QA, and FeTaQA (Chi et al., 2019; Zhu et al., 2021; Nan et al., 2022). Bold numbers denote the best non-oracle results; HTML Oracle uses ground-truth structured HTML.

5 Experiments

Our experiments ask whether current VLMs can produce structured localized differences, how much OCR-derived or oracle structured input helps, and whether auxiliary cell-level warmup improves localization under hard in-domain and out-of-domain settings.

5.1 Main Results on ID-Hard-300

We first evaluate on ID-Hard-300, a complexity-biased subset for cost-controlled API evaluation. Table 2 shows a large gap between task recognition and strict localization: GPT-4o and Qwen3.6-Plus achieve non-trivial task accuracy but near-zero E2E-F1. OCR-to-LLM pipelines improve over direct image prompting but remain far below DiffTable-7B, while strong oracle HTML results confirm that table structure and localization are major bottlenecks.

5.2 Out-of-Domain Generalization

OOD-300 contains examples from SciTSR, TAT-QA, and FeTaQA (Chi et al., 2019; Zhu et al., 2021; Nan et al., 2022), which differ from the in-domain sources in style, layout, and domain. As shown in Table 3, DiffTable-7B retains strong transfer performance, reaching 0.605 E2E-F1 and 0.950 task accuracy, well above zero-shot and OCR-based baselines.

5.3 Full In-Domain Test Results

Because ID-Hard-300 is biased toward complex examples, we also evaluate on the full in-domain test set. Table 4 shows that DiffTable-7B obtains 0.890 E2E-F1 on ID-Full and improves over Direct FT in mIoU and E2E-F1 across ID-Hard, ID-Full, and OOD.

5.4 Effect of Auxiliary OCR-Style Warmup

As shown in Table 4, warmup alone does not solve the benchmark because it lacks paired comparison supervision, but followed by main-task fine-tuning

Training Strategy	ID-Hard-300			ID-Full			OOD-300		
	Task Acc	mIoU	E2E-F1	Task Acc	mIoU	E2E-F1	Task Acc	mIoU	E2E-F1
Warmup only	0.153	0.067	0.000	—	—	—	0.183	0.052	0.000
Direct FT	**0.980**	0.602	0.588	**0.995**	0.752	0.856	**0.950**	0.607	0.530
DiffTable-7B	**0.980**	**0.680**	**0.693**	0.989	**0.804**	**0.890**	**0.950**	**0.642**	**0.605**

Table 4: Ablation and full-set comparison of auxiliary OCR-style warmup. DiffTable-7B first performs OCR-style warmup and then paired-task fine-tuning; dashes indicate settings not evaluated.

it consistently improves mIoU and E2E-F1, suggesting a useful localization prior.

5.5 OCR Bottleneck Analysis

To analyze OCR limitations, we compare OCR HTML outputs against oracle rendering-derived cells on 600 examples and 1200 images. OCR HTML achieves 0.932 position F1 and 0.795 text exact match at matched positions, but cell localization remains much weaker: the final cell-bbox IoU mean is only 0.303 and the IoU ≥ 0.5 rate is 0.280. These failures reflect accumulated table-structure recovery and localization errors rather than text recognition alone. The limitation is amplified for perturbed images with rotation, blur/noise, cropping, or compression, where recovering a fully correct table structure and cell-level bboxes is difficult. Details are in Appendix D.

6 Conclusion

We presented DiffTable, a benchmark for visual table difference detection with structured localization. By applying controlled edits before rendering, DiffTable provides typed difference objects with rendering-derived bboxes and enables evaluation beyond image-level change recognition. Experiments show that zero-shot VLMs and OCR-to-LLM pipelines remain limited under strict end-to-end evaluation: OCR-derived structure helps coarse comparison but is vulnerable to localization errors, especially under visual perturbations. In contrast, DiffTable-7B with auxiliary cell text-and-bbox warmup improves localization-sensitive metrics and achieves strong in-domain and out-of-domain performance. Overall, DiffTable shows that recognizing that two tables differ is far easier than producing an auditable record of where and how they differ.

Limitations

DiffTable uses controlled edits on public table sources to make grounding verifiable: applying a known edit before rendering gives rendering-derived bboxes without relying on OCR predictions or manual box annotation. This design is not meant to claim coverage of all real revision workflows. Each positive pair is generated from one high-level difference family, so the dataset does not include mixed-difference cases where value edits, row/column changes, and cell-span changes occur together. It also does not cover handwritten annotations, formula-level changes, multi-page tables, or tables embedded in complex document layouts. OOD-300 contains three unseen sources, and OCR-pipeline results may change with stronger OCR systems. Future benchmark extensions should include mixed edits, formula-aware table changes, and document-level or multi-page settings.

Ethics Statement

DiffTable is constructed from public table datasets used for research, including table understanding and table QA resources, and all derived examples are generated through controlled edits before rendering. The benchmark is intended for studying visual table comparison and grounded difference detection, not for representing real cases of fraud, misconduct, or document tampering. Because labels are produced from structured sources and browser-rendered table cells, the dataset does not require annotators to inspect private documents or create sensitive personal records.

For anonymous review, we provide a small supplementary demo package containing representative DiffTable examples, prompt templates, controlled-edit construction scripts, and the unified evaluator. The demo is intended to illustrate the data format and evaluation protocol. We intend to release the full DiffTable dataset and code as a free and open research resource for the com-

munity, subject to the licenses and usage terms of the underlying source datasets. Researchers using DiffTable should treat the generated examples as synthetic benchmark data. Systems developed on this benchmark may support auditing, revision tracking, and tampering detection, but their predictions can be wrong and should not be used as fully automated decision makers in high-stakes financial, legal, medical, or scientific workflows without human verification.

References

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-VL Technical Report. *arXiv e-prints*, arXiv:2502.13923.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. Hitab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110.

Zewen Chi, Heyan Huang, Heng-Da Xu, Houjin Yu, Wanxuan Yin, and Xian-Ling Mao. 2019. Complicated table structure recognition. *arXiv preprint arXiv:1908.04729*.

Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, and 1 others. 2025. Paddleocr 3.0 technical report. *arXiv preprint arXiv:2507.05595*.

Zonglin Di, Jing Shi, Yifei Fan, Hao Tan, Alexander Black, John Collomosse, and Yang Liu. 2025. Difftell: A high-quality dataset for describing image manipulation changes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 24580–24590.

Erdong Hu, Longteng Guo, Tongtian Yue, Zijia Zhao, Shuning Xue, and Jing Liu. 2024. Onediff: A generalist model for image difference captioning. In *Proceedings of the Asian Conference on Computer Vision*, pages 2439–2455.

Rajiv Jain and David Doermann. 2013. Visualdiff: Document image verification and change detection. In *2013 12th International Conference on Document Analysis and Recognition*, pages 40–44. IEEE.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2018. Learning to describe differences between pairs of similar images. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4024–4034.

Yuan Liu, Saihui Hou, Saijie Hou, Jiabao Du, Shibei Meng, and Yongzhen Huang. 2025. Omnidiff: A comprehensive benchmark for fine-grained image difference captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21440–21449.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, and 1 others. 2022. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Junbo Niu, Zheng Liu, Zhuangcheng Gu, Bin Wang, Linke Ouyang, Zhiyuan Zhao, Tao Chu, Tianyao He, Fan Wu, Qintong Zhang, and 1 others. 2025. Mineru2. 5: A decoupled vision-language model for efficient high-resolution document parsing. In *The 64th Annual Meeting of the Association for Computational Linguistics–Industry Track*.

Vihang Pancholi, Jainit Sushil Bafna, Tejas Anvekar, Manish Shrivastava, and Vivek Gupta. 2025. Tabxeval: Why this is a bad table? an exhaustive rubric for table evaluation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22913–22934.

Doyoung Park, Naresh Reddy Yarram, Sunjin Kim, Minkyu Kim, Seongho Joe, and Taehee Lee. 2025. Text change detection in multilingual documents using image comparison. In *Proceedings of the Winter Conference on Applications of Computer Vision*, pages 5218–5227.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480.

Abhinandan Kumar Pun, Mohammed Javed, and David S Doermann. 2023. A survey on change detection techniques in document images. *arXiv preprint arXiv:2307.07691*.

Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. 2024. Visual cot: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 37:8612–8642.

Manan Suri, Puneet Mathur, Franck Dernoncourt, Rajiv Jain, Vlad I Morariu, Ramit Sawhney, Preslav Nakov, and Dinesh Manocha. 2024. Docedit-v2: Document structure editing via multimodal llm grounding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15485–15505.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, and 1 others. 2025. Tablebench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 25497–25506.

Wenwen Yu, Zhibo Yang, Jianqiang Wan, Sibo Song, Jun Tang, Wenqing Cheng, Yuliang Liu, and Xiang Bai. 2026. Omniparser v2: Structured-points-of-thought for unified visual text parsing and its generality to multimodal large language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Mingyu Zheng, Xinwei Feng, Qingyi Si, Qiaoqiao She, Zheng Lin, Wenbin Jiang, and Weiping Wang. 2024. Multimodal table understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9102–9124.

Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. 2021. Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 697–706.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2020. Image-based table recognition: data, model, and evaluation. In *European conference on computer vision*, pages 564–580. Springer.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th annual meeting of the Association for Computational Linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pages 3277–3287.

A Dataset Construction Details

DiffTable is generated from structured table sources before rendering. Each source table is first parsed into a structured representation containing cell text, row and column indices, and span information. We then apply one controlled edit operation and render both the original and modified tables as images. Because edits are applied before rendering, the affected regions can be obtained from rendered table-cell positions rather than from OCR predictions or manual annotation.

For value tampering, candidates are visible non-empty cells whose normalized text is not "-". For each positive example, the generator samples between one and four changed cells, limited by the number of available candidates. Numeric perturbation is digit-level: each digit is replaced by a random different digit from 0 to 9, while non-digit characters are preserved. For non-numeric text, one alphabetic character is replaced by a random different alphabetic character while preserving case. If a string has no alphabetic character, the generator appends `x`; empty strings are mapped to `X`.

For structural changes, row and column candidates exclude table boundaries and avoid rows or columns that would conflict with existing spans. Row and column additions are constructed by reversing a deletion operation: one real row or column is removed from one side, and the unmodified source table is used as the other side. Thus, inserted regions contain natural table content from the source table rather than randomly synthesized filler. The target box is the rendered union box of the whole affected row or column, and the missing side is represented by `null`.

For cell-span changes, the generator selects non-overlapping adjacent cells that are visible, non-header cells with unit rowspan and colspan. It samples up to three horizontal or vertical merges per example, always treating Table A as the split side and Table B as the merged side for the `cell_merged` label. Table A keeps the original cells and marks all affected cells with a shared merge identifier, while Table B removes the covered cell and assigns rowspan or colspan to the anchor cell. The Table-A box is the union box over the affected split cells, and the Table-B box is the rendered box of the merged cell. No-difference examples are generated by pairing a table with itself; they contain no semantic table-content edit, although rendering and image-space perturbations may still make the two images visually non-identical. These perturbations are nuisance variation rather than positive difference labels.

Source filtering is applied before editing. The generators require parseable tables with minimum and maximum row, column, cell-count, text-length, and aspect-ratio constraints. For the OOD builder, the default constraints keep tables with 3–34 rows, 3–14 columns, 9–220 grid cells, and rendered aspect ratio between 0.15 and 6.0; matrix-based sources are normalized and long cell text is truncated to 80 characters before rendering. The source-specific in-domain generators use analogous soft filters and additionally check that rendered boxes remain inside image bounds.

Rendering introduces visual diversity before annotation export. In-domain examples use randomized CSS styles, paper-like background colors, random margins, and image-space transforms; the modified side may be rotated within $\pm 6^\circ$, translated by up to 40 pixels, cropped, and lightly blurred or darkened. OOD examples use broader scene templates, table styles, and font families. They further apply low, medium, and high perturbation profiles with maximum rotations of 1.6, 3.2, and 5.0 degrees, translations up to 6, 12, and 20 pixels, and JPEG qualities of 90–96, 82–92, and 76–88, respectively.

The final OOD images receive an additional photo/scanned-paper post-processing step. This step applies rotation in $[-8.5^\circ, 8.5^\circ]$, anisotropic scaling, random cropping, textured backgrounds, drop shadows, contrast/brightness/color/sharpness changes, grayscale or tint effects, low-resolution resampling, blur, noise, stripe artifacts, and JPEG recompression with quality 46–78. All cell boxes, answer boxes, and oracle HTML boxes are transformed through the same geometry so that the final labels remain aligned with the final images. For OOD source selection, we draw from SciTSR, TAT-QA, and FeTaQA (Chi et al., 2019; Zhu et al., 2021; Nan et al., 2022): SciTSR prioritizes native span-aware complex tables, TAT-QA uses converted financial question-answering tables, and FeTaQA uses test-only hard candidates that prefer large, wide, long-text, or extreme-aspect tables.

Source-table leakage checks confirm zero overlap between training and evaluation splits. We also run automated integrity checks for missing images, invalid JSON, out-of-range boxes, parsing failures, and related split-integrity issues.

Category	Fine-grained type	Required boxes	Required text
Value tampering	`value_tampering`	`box_a`, `box_b`	`text_a`, `text_b`
Structural insertion	`row_added`, `col_added`	`box_a = null`, `box_b`	Not required
Structural deletion	`row_deleted`, `col_deleted`	`box_a`, `box_b = null`	Not required
Cell-span change	`cell_merged`	`box_a`, `box_b`	Not required
No difference	–	Empty list	Not required

Table 5: Output requirements for each DiffTable category. Missing sides are represented as `null`, and no-difference examples require an empty JSON list.

B Additional Dataset Statistics

Table 6 reports additional statistics for the evaluation splits. ID-Hard-300 is intentionally biased toward larger and more object-dense examples, whereas ID-Full reflects the average in-domain distribution.

C Evaluation Protocol

We first extract the first JSON array from each model output; extraction failure yields an empty list. Non-dictionary array entries are ignored because they cannot represent difference objects. Every parsed dictionary is otherwise retained in the prediction multiset. Objects with unsupported or missing `diff_type` are marked invalid: they cannot enter candidate matching, but they still count as unmatched false positives in object-level metrics.

Boxes are parsed as $[x_1, y_1, x_2, y_2]$ rectangles that are expected to use normalized image coordinates. The evaluator accepts a side box when it is a numeric four-tuple with nonzero area after endpoint ordering. Malformed or degenerate boxes are treated as missing on that side; out-of-range coordinates are not clipped and fail localization unless they overlap the corresponding ground-truth box. The prediction object itself remains in the prediction multiset. Task labels are recomputed from the preprocessed prediction and ground-truth JSON before scoring: an empty prediction is labeled no difference, whereas a non-empty prediction with mixed, unsupported, or missing types receives an invalid task label and is counted as incorrect. Thus invalid or malformed prediction dictionaries are not silently deleted from object-level precision; they either fail task labeling, become unmatched false positives, or fail the localization predicate through missing boxes.

Task accuracy maps each prediction to one of four high-level labels: value tampering, structural change, cell-span change, or no difference. A non-empty prediction with mixed or unsupported types is counted as incorrect. Let N be the number of evaluated examples, t_i be the ground-truth high-level task label, and $\hat{t}_i$ be the predicted task label. Task accuracy is computed as

$$\mathrm{TaskAcc} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{I}[\hat{t}_i = t_i], \tag{C.1}$$

where $\mathrm{I}[\cdot]$ is an indicator function and $|\cdot|$ denotes set cardinality. Macro-F1 is computed over the four high-level task labels:

$$\mathrm{MacroF1} = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \frac{2P_cR_c}{P_c + R_c}, \tag{C.2}$$

where $\mathcal{C}$ contains value tampering, structural change, cell-span change, and no difference, and P_c and R_c are the sample-level precision and recall for class c. If $P_c + R_c = 0$, the class-F1 term in Equation C.2 is set to zero.

Object-level matching is performed within the same fine-grained difference type. Let p denote a predicted object, g denote a ground-truth object, and $s \in \{a, b\}$ denote the Table A or Table B side. For each candidate prediction-ground-truth pair, we compute IoU on each side where both the prediction and ground truth contain a valid box:

$$\mathrm{IoU}(b, \hat{b}) = \frac{\mathrm{area}(b \cap \hat{b})}{\mathrm{area}(b \cup \hat{b})}. \tag{C.3}$$

Here b and $\hat{b}$ are two valid rectangles in the same side-specific coordinate system, and $\mathrm{area}(\cdot)$ denotes rectangle area. Let p_s and g_s be the boxes of p and g on side s, respectively. Let $S(p, g)$ be the set of sides with valid boxes in both p and g. For non-empty $S(p, g)$, define the average side IoU

$$A(p, g) = \frac{1}{|S(p, g)|}\sum_{s\in S(p,g)} \mathrm{IoU}(p_s, g_s). \tag{C.4}$$

The matching score is

$$\mathrm{pairIoU}(p, g) = \begin{cases} A(p, g), & S(p, g) \neq \emptyset, \\ 0, & S(p, g) = \emptyset. \end{cases} \tag{C.5}$$

Split	Samples	GT Objects	Avg. GT Obj. (+)	Avg. Cells	Avg. Text Chars
ID-Full Test	3,771	6,266	1.85	63	489
ID-Hard-300	300	563	2.09	155	1,160
OOD-300	300	495	1.83	81	725

Table 6: Additional object and input-complexity statistics. Avg. GT Obj. (+) is computed over positive examples only.

Input feature	ID-Full	ID-Hard-300	Ratio
Avg. cells	63	155	2.47×
Avg. rows	12.3	26.0	2.11×
Avg. columns	5.5	6.9	1.25×
Avg. text chars	489	1,160	2.37×
Avg. image width	847	1,029	1.21×
Avg. image height	583	1,075	1.84×

Table 7: Input-complexity comparison between ID-Full and ID-Hard-300. Ratio is computed as ID-Hard-300 divided by ID-Full.

All same-type candidate pairs are sorted by pairIoU and matched greedily under a one-to-one constraint. Same-type pairs with $\mathrm{pairIoU}(p, g) = 0$ may therefore enter $\mathcal{M}$ with zero score; they lower mIoU and cannot satisfy the localization or end-to-end true-positive predicates. Consequently, $|\mathcal{M}|$ counts greedy same-type matched pairs rather than only positive-overlap pairs.

Let $\mathcal{M}$ be the set of matched prediction-ground-truth object pairs after greedy matching. Mean IoU is computed over matched object pairs:

$$\mathrm{mIoU} = \frac{1}{|\mathcal{M}|} \sum_{(p,g)\in\mathcal{M}} \mathrm{pairIoU}(p, g). \tag{C.6}$$

The IoU threshold metric measures the fraction of matched object pairs satisfying the type-specific localization predicate $L_{0.5}(p, g)$:

$$\mathrm{IoU}_{0.5} = \frac{1}{|\mathcal{M}|} \sum_{(p,g)\in\mathcal{M}} \mathrm{I}[L_{0.5}(p, g)]. \tag{C.7}$$

For value tampering and cell-span changes, $L_{0.5}$ requires both sides to have IoU at least 0.5; a missing side contributes zero. For structural changes, $L_{0.5}$ requires correct null-side alignment and IoU at least 0.5 on the non-null side. If there are no matched pairs, mIoU in Equation C.6 and IoU@0.5 in Equation C.7 are set to zero.

A matched pair is counted as an end-to-end true positive only if all required conditions are satisfied. Value tampering requires correct type, IoU ≥ 0.5 on both sides, and exact matches for `text_a` and `text_b`. Cell-span changes require correct type and IoU ≥ 0.5 on both sides. Structural changes require correct fine-grained type, correct null-side alignment, and IoU ≥ 0.5 on the non-null side. No-difference examples contain no ground-truth objects; an empty prediction affects only sample-level task accuracy, while any non-empty prediction contributes false-positive objects.

For value tampering, let p_a^{text} and p_b^{text} denote the predicted `text_a` and `text_b`, and let g_a^{text} and g_b^{text} denote the corresponding ground-truth texts. Define side-specific exact-match indicators $T_a(p, g)$ and $T_b(p, g)$, and let $T(p, g)$ be 1 when both edited texts match exactly and 0 otherwise:

$$\begin{aligned} T_a(p, g) &= \mathrm{I}[p_a^{\mathrm{text}} = g_a^{\mathrm{text}}], \\ T_b(p, g) &= \mathrm{I}[p_b^{\mathrm{text}} = g_b^{\mathrm{text}}], \\ T(p, g) &= T_a(p, g)T_b(p, g). \end{aligned} \tag{C.8}$$

Text-EM is computed over matched value-tampering pairs:

$$\mathrm{TextEM} = \frac{1}{|\mathcal{M}_v|} \sum_{(p,g)\in\mathcal{M}_v} T(p, g), \tag{C.9}$$

where $\mathcal{M}_v$ denotes matched value-tampering pairs. Text comparison strips leading and trailing whitespace in the evaluator; some prediction wrappers additionally compact whitespace before writing outputs. If no value-tampering pair exists for a method or setting, Text-EM is not applicable and is reported with a dash.

Let $\mathcal{P}$ be all parsed prediction dictionaries retained by the pre-evaluation parsing protocol, including invalid-type objects and objects predicted

on no-difference examples, and let $\mathcal{G}$ be all ground-truth objects. No-difference examples contribute no objects to $\mathcal{G}$. Let $E(p, g)$ indicate whether a matched pair satisfies all end-to-end requirements for its type. For compactness, write TP, FP, and FN for end-to-end true-positive, false-positive, and false-negative counts:

$$\begin{aligned} \mathrm{TP} &= \sum_{(p,g)\in\mathcal{M}} E(p, g), \\ \mathrm{FP} &= |\mathcal{P}| - \mathrm{TP}, \\ \mathrm{FN} &= |\mathcal{G}| - \mathrm{TP}. \end{aligned} \tag{C.10}$$

End-to-end precision, recall, and F1 are then

$$\begin{aligned} P_{\mathrm{e2e}} &= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \\ R_{\mathrm{e2e}} &= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \\ F1_{\mathrm{e2e}} &= \frac{2P_{\mathrm{e2e}}R_{\mathrm{e2e}}}{P_{\mathrm{e2e}} + R_{\mathrm{e2e}}}. \end{aligned} \tag{C.11}$$

The zero-denominator cases are handled explicitly: if $\mathrm{TP} + \mathrm{FP} = 0$, then $P_{\mathrm{e2e}} = 0$; if $\mathrm{TP} + \mathrm{FN} = 0$, then $R_{\mathrm{e2e}} = 0$; and if $P_{\mathrm{e2e}} + R_{\mathrm{e2e}} = 0$, then $F1_{\mathrm{e2e}} = 0$.

As a boundary-case example, consider a one-example corpus whose ground truth is no difference. If the prediction is empty, then $\hat{t} = t$, so TaskAcc receives one correct sample, while $|\mathcal{P}| = |\mathcal{G}| = \mathrm{TP} = \mathrm{FP} = \mathrm{FN} = 0$; mIoU, IoU@0.5, E2E-P, E2E-R, and E2E-F1 are all zero, and Text-EM is reported as a dash. This no-difference success is credited by TaskAcc, not by creating an artificial object-level true positive. If the same no-difference example has one retained predicted object, then TaskAcc is incorrect, $|\mathcal{P}| = 1$, $|\mathcal{G}| = 0$, $\mathrm{TP} = 0$, $\mathrm{FP} = 1$, and $\mathrm{FN} = 0$, so the end-to-end metrics remain zero.

D OCR Bottleneck Analysis

We compare OCR HTML outputs against oracle rendering-derived cells on 600 examples and 1,200 images. The OCR pipeline achieves high position-level matching but much weaker final cell-box overlap. This explains why OCR-to-LLM pipelines improve over zero-shot image prompting but remain weak under strict object-level evaluation.

The main failure mode is localization and table-structure recovery rather than only text recognition. Although OCR recovers many cell texts, its derived cell boxes are not consistently aligned with the rendering-derived target cells; errors can arise from missed or merged cells, imperfect row/column grouping, and geometric inaccuracies under visual perturbations. These localization errors propagate into false positives and false negatives in end-to-end matching.

Metric	Value
Position F1	0.932
Text exact match	0.795
Mean cell-box IoU	0.303
Cell-box IoU ≥ 0.5 rate	0.280

Table 8: OCR HTML quality analysis against rendering-derived oracle cells on 600 examples and 1,200 images. The gap between position matching and cell-box IoU highlights localization error in OCR-derived inputs.

E Additional Experimental Results

Table 9 reports fine-grained subtype accuracy and the task-level confusion matrix for DiffTable-7B. The model performs strongly on value tampering and cell-span changes, while row deletion and column deletion are relatively more difficult in some settings.

Table 10 shows that most remaining high-level errors are missed positive differences rather than false alarms on no-difference samples. At the object level, the residual unmatched count indicates that strict localization and text recovery still account for a meaningful share of end-to-end failures even when the high-level task label is correct.

F Prompts and Training Details

We use a shared JSON output schema but different prompt families for different evaluation settings. Figure 3 summarizes the fine-tuned image inference prompt, direct-image API prompt, oracle-HTML prompt, and OCR-HTML prompt. The figure gives faithful English renderings of the executable Chinese prompts used in the test scripts.

DiffTable-7B is initialized from Qwen2.5-VL-7B-Instruct (Bai et al., 2025); the base model files correspond to the Hugging Face snapshot prefix `cc594898`. The final model uses two stages. The first stage is an auxiliary OCR-style warmup on 7,279 single-table examples, where the model predicts rendering-derived cell texts and bboxes from one table image. The second stage fine-tunes the model on 30,321 paired DiffTable examples. We use QLoRA with 4-bit bnb quantization, LoRA

ID-Hard-300 Subtype Accuracy				OOD-300 Subtype Accuracy				ID-Hard-300 Confusion				
Subtype	**GT**	**Match.**	**Acc.**	**Subtype**	**GT**	**Match.**	**Acc.**	**GT/Pred.**	**VT**	**SC**	**CM**	**ND**
value_tampering	231	217	0.939	value_tampering	221	204	0.923	VT	90	0	0	0
cell_merged	242	228	0.942	cell_merged	172	160	0.930	SC	1	85	0	4
col_added	20	20	1.000	col_added	28	28	1.000	CM	0	0	89	1
col_deleted	27	25	0.926	col_deleted	22	17	0.773	ND	0	0	0	30
row_added	20	18	0.900	row_added	28	27	0.964	–	–	–	–	–
row_deleted	23	21	0.913	row_deleted	24	20	0.833	–	–	–	–	–

Table 9: Fine-grained subtype accuracy and task-level confusion analysis for DiffTable-7B. Match. counts correctly matched ground-truth subtype objects; the confusion matrix reports high-level task predictions on ID-Hard-300.

Error source	**Unit**	**Count**
Wrong non-ND high-level type	sample	1
Positive predicted as no-diff	sample	5
No-diff false alarm	sample	0
Positive objects not matched	object	34

Table 10: Error-source summary for DiffTable-7B on ID-Hard-300, derived from Table 9. "Positive objects not matched" is the residual between ground-truth subtype objects and correctly matched subtype objects ($563 - 529$), and may include missing objects, localization failures, or text mismatches.

rank 32, alpha 64, dropout 0.05, target modules set to all supported linear/projection modules, bf16 training, a cosine learning-rate schedule, learning rate 1×10^{-4}, cutoff length 4096, and two epochs for paired-task fine-tuning.

The reported training run used four NVIDIA GeForce RTX 3090 GPUs with 24GB memory each. The auxiliary warmup used per-device batch size 1 with gradient accumulation 4, giving an effective total batch size of 16 across the four GPUs; it ran for one epoch, 455 optimizer steps, and about 2.2 hours of logged trainer runtime. The paired-task stage used per-device batch size 2 with gradient accumulation 2, again giving an effective total batch size of 16; it ran for two epochs, 3,792 optimizer steps, and about 16.1 hours of accumulated logged trainer runtime across resumed runs. The LoRA adapters contain 103,043,072 trainable parameters.

```
OUTPUT_SCHEMA = [
  {box_a, text_a, box_b, text_b, diff_type}      value_tampering
  {box_a, box_b, diff_type}                      row/column add-delete or cell_merged
  []                                             no_diff
]
```

LoRA image evaluation

image_input: two image tokens are passed through the VLM processor.

prompt_behavior: the evaluator reads messages[0].content from each test sample, removes literal Picture 1 / Picture 2 lines and <image> placeholders, then appends the remaining task instruction after the two image tokens.

value_tampering_variant: Compare two table images, locate cells whose text was modified, and output a JSON array with fields box_a, text_a, box_b, text_b, diff_type. Coordinates use normalized [0,1000] boxes; diff_type is only value_tampering; output [] if there is no content difference.

struct_change_variant: Compare two table images, decide whether a row or column was added/deleted, and localize the changed region. Output fields box_a, box_b, diff_type; use null for the absent side; diff_type is row_added, row_deleted, col_added, or col_deleted; output [] if there is no structural difference.

cell_merged_variant: Compare two table images, decide whether cell merging occurred, and localize the changed region. Output fields box_a, box_b, diff_type; use null for the absent side; diff_type is only cell_merged; output [] if there is no such structure difference.

decoding: greedy generation with do_sample=False and max_new_tokens=512.

Direct-image API evaluation

system: You receive two table images: Picture 1 is Table A and Picture 2 is Table B. Compare the tables, localize visible differences, and output only a strict JSON array.

coordinate_rules: Every box is [x1,y1,x2,y2], normalized to the width and height of its own image, in the range 0..1000. Require 0<=x1<x2<=1000 and 0<=y1<y2<=1000. Do not output pixel coordinates, percentages, decimals, or 0..1 ratios.

localization_rule: The two images may have different crops, scales, aspect ratios, and table positions. Observe Picture 1 and Picture 2 independently; do not copy coordinates across images or assume corresponding cells share coordinates.

allowed_tasks: value_tampering with box_a,text_a,box_b,text_b,diff_type; struct_change with row_added,row_deleted,col_added,col_deleted and box_a,box_b,diff_type; cell_merged with box_a,box_b,diff_type; no_diff as [].

decision_order: First compare row/column counts, horizontal and vertical grid lines, and local cell boundaries. Whole-row/column addition/deletion is struct_change, not value_tampering. Local merge/split or internal boundary changes are cell_merged, not value_tampering. Judge value_tampering only when the grid topology is basically aligned. Do not mix task families in one output array.

ignore: minor rendering differences, font weight, pixel noise, crop edges, whitespace shifts, global scaling, and translation; focus on visible text and table structure.

box_rules: For value_tampering, box the complete table cell and include text_a/text_b. For row_added/row_deleted, box the whole visible row. For col_added/col_deleted, box the whole visible column. For added regions use box_a=null; for deleted regions use box_b=null. For cell_merged, box one local enclosing region for the affected merge/split area and do not expand it to a full row/column.

output_format: JSON array only, without Markdown, code fences, explanation, or natural-language prefix/suffix. Output at most max_items clearest differences.

HTML oracle evaluation

input: Compact HTML for Table A and Table B after removing CSS/head/body/meta. Each rendered cell keeps , , and .

use: <table>/<tr>/<td> hierarchy, row/col, rowspan/colspan, cell text, and bbox_norm. bbox_norm is an integer [x1,y1,x2,y2] box normalized to each image in 0..1000.

prior: Each table pair has at most one difference family; do not mix value_tampering, struct_change, and cell_merged. If multiple candidates appear, first decide that they belong to the same type.

bbox_rule: Do not estimate coordinates visually. Compute every output box directly from relevant HTML cell bbox_norm values. For a multi-cell region, collect related cells and output the minimum enclosing rectangle [min(x1),min(y1),max(x2),max(y2)]. Do not split one structural region into scattered small boxes.

tasks: value_tampering means same structure but different corresponding cell text. struct_change means whole row/column add/delete; use the union box of cells covering the target row/column and null on the absent side. cell_merged means local merge/split changing grid boundaries or rowspan/colspan; one local region may have one merged cell on one side and multiple split cells on the other side.

priority: row/column add-delete first, then cell merge/split, then text modification when the structure is consistent. Return [] if there is no difference. Append Table A/B structure summaries and compact HTML blocks.

OCR HTML evaluation

input: OCR-derived compact HTML for Table A and Table B, not manually annotated true table structure. Each OCR text block keeps , , and . Row/col indices are approximate layout indices inferred from bbox positions, not reliable true cell-grid indices.

use: OCR block spatial positions, neighboring block order, and text content. bbox_norm is already normalized to each image in 0..1000.

prior: Each table pair has at most one difference family; do not mix value_tampering, struct_change, and cell_merged.

noise_rule: Do not treat pure OCR recognition noise as real differences. Prefer ignoring spacing, punctuation, full-/half-width changes, case, O/0, l/1, day/0, minor missing characters, brackets, and hyphens unless there is a stable structural or semantic numeric change.

bbox_rule: Never visually re-estimate coordinates. Derive boxes from related OCR block bbox_norm values and merge them with the minimum enclosing rectangle.

tasks: value_tampering requires clear text modification in corresponding regions. struct_change uses the enclosing box of the added/deleted row band or column band and should explain large neighboring-block shifts rather than splitting them into many text edits. cell_merged is used only when a local region is clearly one large block on one side and multiple smaller blocks on the other, not ordinary OCR splitting or line-break noise.

priority: row/column add-delete first, clear text modification second, local merge/split last. Return [] if no difference. Append Table A/B structure summaries and compact OCR HTML blocks.

Figure 3: Prompt template families used in DiffTable evaluation. The fine-tuned model uses task-specific prompts stored in the test data and injects the two image tokens through the VLM processor; direct-image API baselines use a longer system prompt with explicit coordinate, decision-priority, and boxing rules; HTML-oracle and OCR-HTML baselines use structured-input prompts with the same JSON output schema.