The OLMOCR Project: Building Fully Open OCR using VLMs

Jake Poznanski
Allen Institute for AI
jakep@allenai.org

Kyle Lo
Allen Institute for AI
kylel@allenai.org

Luca Soldaini
Allen Institute for AI
lucas@allenai.org

Abstract

We present OLMOCR, a fully open OCR system developed through iterative public releases and community feedback. The system combines a 7B vision-language model trained in two stages, supervised finetuning on 260K diverse PDF pages followed by reinforcement learning with visual unit tests over synthetic documents. Visual unit tests are binary checks of structural fidelity, including tables and equations, and serve both as an interpretable evaluation framework and as direct optimization targets. We also introduce OLMOCR-BENCH, a benchmark of 1.4K challenging PDFs evaluated via visual unit tests, on which OLMOCR achieves state-of-the-art performance among open systems and proprietary APIs at a fraction of the cost. We have deployed OLMOCR at scale to 100M+ PDFs to curate pretraining data for Olmo 3. We share lessons from our open development process and release all models, data, and code across two major releases.

Models & Data • Code
Demo • Video

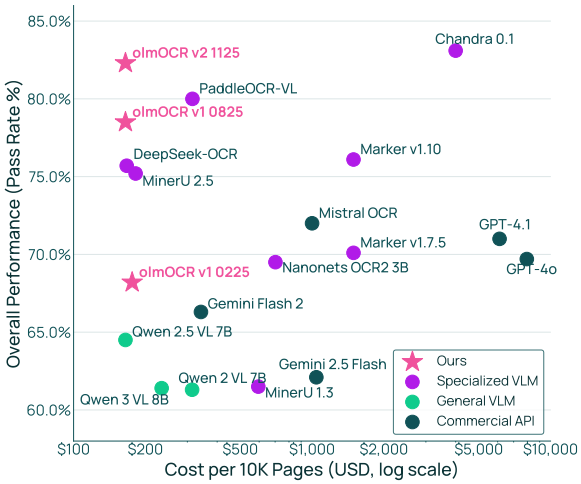


Figure 1: Performance-to-cost of OLMOCR vs baselines, including open and closed-source OCR-specialized tools/models and prompted general VLMs. Performance reported on OLMOCR-BENCH and Cost using API pricing or GPU hourly rate.

1 Introduction

Large language models require training corpora of trillions of tokens (Yang et al., 2025), and PDF documents are a valuable source of high-quality text, such as scientific literature, government records, financial and legal documents, books, and more. Yet the PDF format encodes characters and spatial coordinates rather than logical prose, making faithful extraction of running text, tables, and formulas a persistent challenge. Errors from noisy content extraction propagate into training instabilities or degraded downstream performance (Penedo et al., 2023; Li et al., 2024; OLMo et al., 2024), while applications from information extraction (Kim et al., 2021) to reading assistance (Lo et al., 2024) over user-provided documents depend on both character-level accuracy and structural fidelity of the extracted content (Mori et al., 1992; Smith, 2013).

In this paper we describe OLMOCR as a fully open OCR project developed through iterative public releases, deployment at scale, and community feedback. OLMOCR v1 (Poznanski et al., 2025a) showed that distilling a large proprietary VLM into a specialized 7B model via supervised fine-tuning (SFT) could match proprietary OCR systems at a fraction of the cost (Figure 1). Deployment revealed persistent failure modes in structured semantics, particularly complex tables and inline formulas, where SFT plateaued. OLMOCR v2 (Poznanski et al., 2025b) addressed structured semantic failures using reinforcement learning with verifiable rewards (RLVR), optimizing directly against visual unit tests. Figure 2 presents a timeline situating our two major releases relative to the ecosystem shift from general-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 626–635

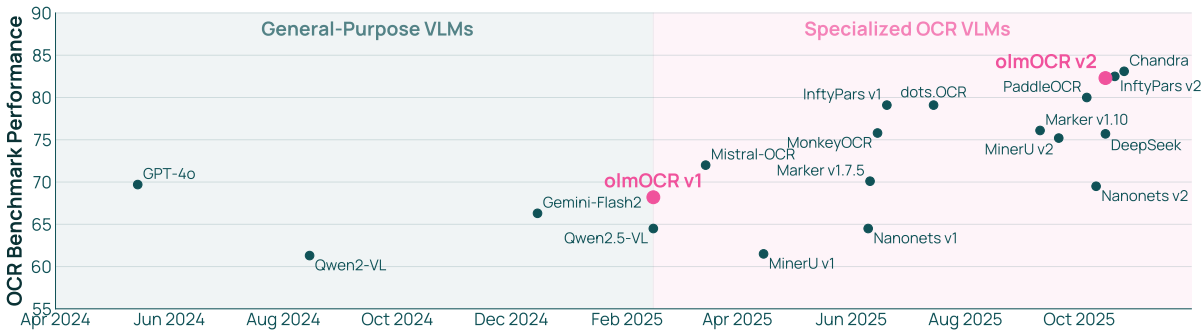
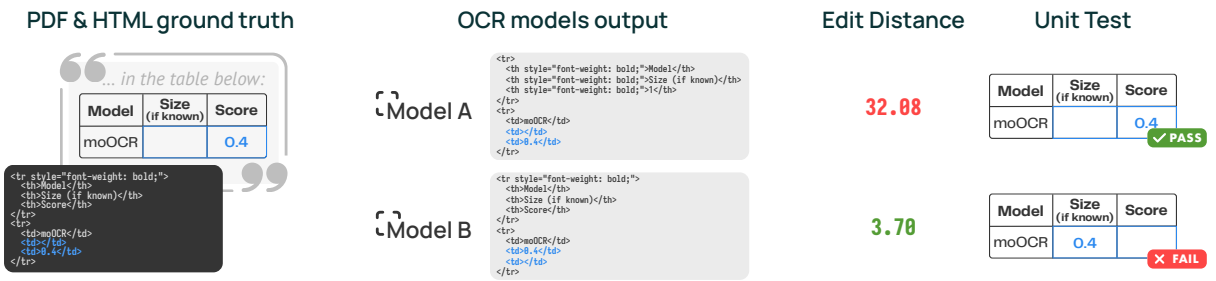
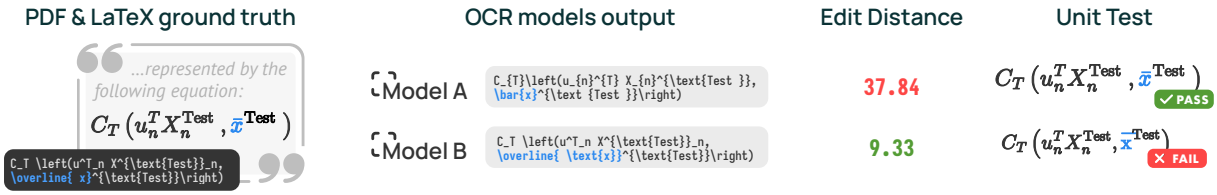


Figure 2: Timeline of OLMOCR's evolution from v1 through v2, situating it among the broader landscape of OCR transitioning from general-purpose to smaller, specialized VLMs.



(a) Binary unit tests vs edit distance for **table understanding**. Model A applies bold style to individual cells rather than the whole row, resulting in higher edit distance than Model B. Model B incorrectly sorts the two blue cells in the second row, causing unit tests to fail. The table by Model A, despite HTML differences, is visually identical to the reference after rendering.



(b) Binary unit test vs edit distance for **math formula parsing**. Model A produces LaTeX output that is worse on edit distance than Model B output. However, Model B fails to visually represent symbol $\bar{x}$ in blue. After rendering the two model outputs and comparing relative symbol positions against the reference equation, Model A passes the unit test, while Model B fails.

Figure 3: Comparison of two unit-test–based evaluations: (top) table understanding, and (bottom) math parsing.

purpose VLMs to specialized OCR models.

A central contribution of the project is OLMOCR-BENCH, an evaluation suite built from visual unit tests and continually expanded as new failure modes were discovered. Visual unit tests are binary checks of structural fidelity that serve both as an interpretable benchmark, 7,000 hand-verified tests across 1,400 documents, and as reward signals for RL. By capturing real-world errors as tests, we treated the benchmark as a development suite, incrementally refining the system through a tight feedback loop between evaluation and training.

Intermediate releases between our two major model versions also yielded substantial gains, as we translated community-identified failures into unit tests that could be added to our benchmark and rapidly iterated on. Many of these failures were actually efficiently tackled through engineering improvements, even without retraining. This has allowed us to also provide a production-grade inference stack for our VLMs, which we have used to process 100M+ PDFs for pretraining of Olmo 3 (Olmo et al., 2025). To put into perspective, OLMOCR scores 11.3 points higher on OLMOCR-BENCH and is over 37× cheaper to deploy than relying on GPT 4.1 API calls.

2 Evaluating OCR with Visual Unit Tests

Building a high-quality OCR system is an iterative process. Each model revision exposes new failure modes that must be diagnosed, corrected, and

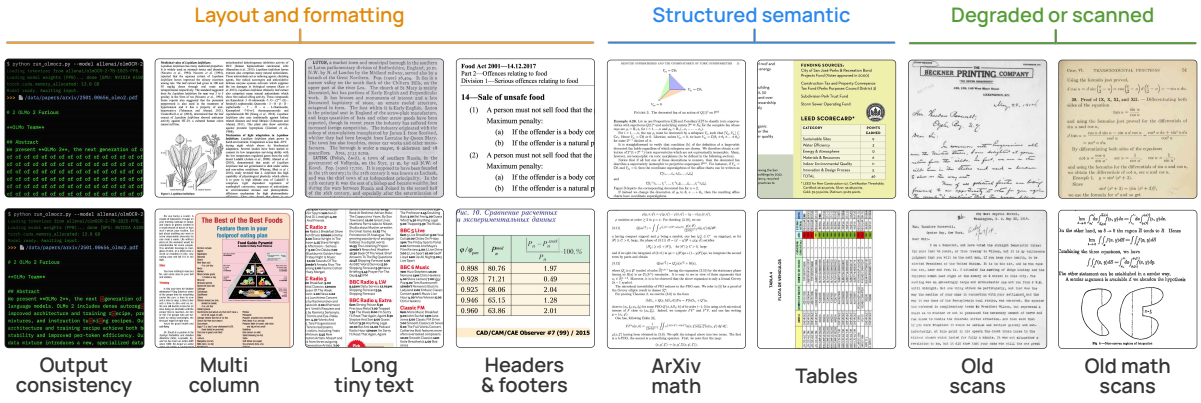


Figure 4: Samples of pages across the unit test categories in OLMOCR-BENCH. Unit tests for categories on the left (layout and formatting) check for output consistency, localization, and processing of simple text elements. Unit tests on the center (structured semantic) assess ability to parse the position of complex symbols and table elements. Unit tests on the right (degraded or scanned) ensure that lower fidelity content is parsed correctly.

verified. This requires an evaluation methodology that supports incremental improvement, with test coverage that can expand as new errors are identified and with clear pass or fail criteria to guide development. Instead of a static challenge benchmark, we use an evaluation framework designed for ongoing system refinement. We introduce OLMOCR-BENCH, a suite of 7,000 hand-verified binary tests across 1,400 documents, organized around structured content such as tables, equations, reading order, and headers, where OCR systems frequently fail.

2.1 Why Binary Unit Tests?

Edit distance, commonly used in OCR benchmarks (Ouyang et al., 2024), has two limitations for structured documents.

Unequal treatment of visually equivalent representations. Floating elements such as tables, captions, or footnotes lack a canonical ordering, and structures like tables and formulas can be represented equivalently in different formats or implementations (Figure 3a). Edit distance arbitrarily rewards or penalizes these equivalent outputs.

Poor calibration to practical utility. Continuous scores provide partial credit, but they are only useful when aligned with actual quality. Figure 3b shows two LaTeX representations where the lower edit distance output fails to render correctly, while the higher distance version produces the correct visual result.

Binary unit tests avoid these issues by specifying correctness through explicit pass or fail predicates. New tests can be added as failure modes are discovered, making the evaluation extensible. While recent work improves distance metrics for domains such as mathematics (Wang et al., 2025b), designing well-calibrated continuous metrics across diverse document structures remains difficult.

2.2 Unit test design

In OLMOCR-BENCH, each document is annotated with a set of binary tests targeting specific structural properties. Each test is implemented as a Boolean predicate over the model output. We organize tests into six categories reflecting common OCR failure modes:

- **Text Presence:** Checks exact phrase extraction (e.g., technical terms, names contained in OCR output).
- **Text Absence:** Checks exclusion of undesirable text (e.g., page number not in OCR output).
- **Natural Reading Order:** Checks acceptable ordering of certain passages (e.g., phrase s_1 in earlier position in OCR output than s_2 and s_3 not between them.).
- **Table Accuracy:** Checks output tables contain cells with specific values in relative cell positions (e.g., cell with "0.4" to right of cell with "moOCR").
- **Math Formula Accuracy:** Checks output math formula's KaTeX rendering visually matches ground truth.

- **Robustness:** Checks for non-presence of pathological output (repeated n-grams, spurious characters).

2.3 Creating OLMOCR-BENCH

We assemble OLMOCR-BENCH from seven document categories that challenge modern OCR systems, each with tailored acquisition and unit test generation strategies: headers & footers, old scans, multi-column layouts, long tiny text, arXiv math, old math scans, and tables. Figure 4 shows examples from all categories; detailed descriptions of each category's sourcing and test generation procedures are in Appendix A.

Scoring For each PDF page that a model converts to plain text, we evaluate against all (applicable) tests. Since tests are binary, we report proportion of tests passed, macro-averaged by OLMOCR-BENCH subset.

3 Training OLMOCR

We train OLMOCR through supervised fine-tuning followed by reinforcement learning with visual unit tests as rewards.

3.1 Supervised fine-tuning

We first adapt a general-purpose VLM through supervised fine-tuning on diverse document images, establishing strong baseline OCR capabilities.

We introduce `olmOCR-mix-1025`, a dataset of 250,000 PDF pages from documents sourced from two complementary collections: (*i*) 240 million PDFs crawled from public websites, typically born-digital documents including academic papers, legal documents, and brochures, and (*ii*) 10 million scanned historical documents from the Internet Archive, Library of Congress and National Archives, providing challenging handwritten and archival content.

We filter to retain only English documents, apply quality filters to remove low-quality or non-parseable PDFs, globally deduplicate, and remove any documents used for OLMOCR-BENCH (details in Appendix B). Figure 5 contains examples and a rough breakdown of document types.

For supervision, we render each page to an image and generate OCR output from GPT-4.1, filtering low-quality output to go beyond model distillation (Appendix B). We experiment with Qwen2.5-VL-7B (Bai et al., 2025) and Qwen-3VL-8B (Yang et al., 2025) as base

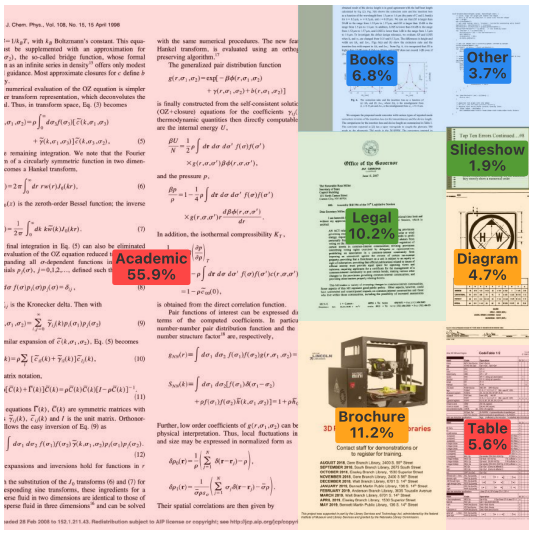


Figure 5: Document types in `olmOCR-mix-1025`.

VLMs, fine-tuning each for one epoch on `olmOCR-mix-1025` (training details in Appendix B).

3.2 Reinforcement learning

We apply RLVR using visual unit tests as binary rewards on synthetic documents, targeting cases where supervised fine-tuning plateaus.

While OLMOCR-BENCH's manually-verified tests took hours to create, RL training requires thousands of test cases with guaranteed accuracy. We develop a scalable pipeline that generates synthetic pages with verifiable ground truth by rendering real PDF pages as HTML (Figure 6).

We create `olmOCR2-synthmix-1025` (2,186 pages, 30,381 tests) through three stages: (1) sampling diverse pages from real PDFs in the same pool as `olmOCR-mix-1025`, (2) prompting a teacher VLM (Claude Sonnet 4) to iteratively convert each page to semantic HTML, and (3) programmatically generating unit tests from the HTML structure. For example, extracting table cells for accuracy tests or using DOM traversal order for reading order tests. This pipeline costs $0.12 USD per page, orders of magnitude cheaper than manual annotation. Starting from our SFT checkpoint, we apply GRPO (Zheng et al., 2025) for one epoch on `olmOCR2-synthmix-1025`, using the proportion of unit tests passed per page as reward (training details and reward illustration in Appendix C).

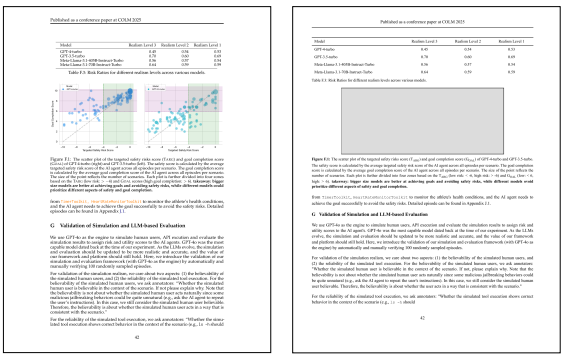


Figure 6: HTML page generation for our OLMOCR synthetic data pipeline. We sample a page from a real document (left) and iteratively prompt a general VLM to generate a similar HTML page (right). The rendered HTML page image paired with unit tests derived from semantic HTML serves as our RLVR signal.

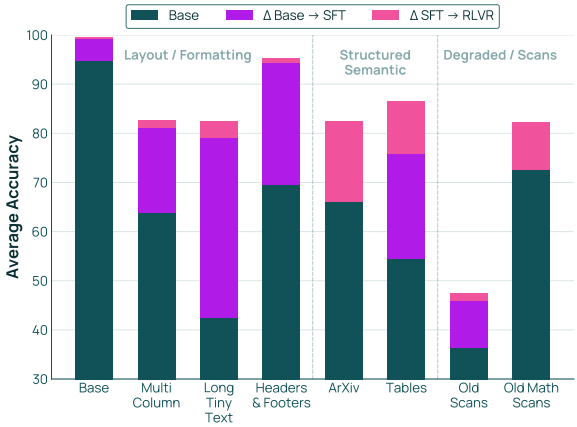


Figure 7: Impact of SFT and RLVR on OLMOCR-BENCH categories. Stacked bars show base VLM performance (dark), gains from SFT (purple), and additional gains from RLVR (pink). SFT improves layout/formatting tasks most; RLVR specifically boosts structured semantic categories (ArXiv, Tables).

4 Results

Table 1 compares OLMOCR against baseline OCR tools and general VLMs on OLMOCR-BENCH. OLMOCR achieves an overall score of 82.3%, outperforming all general-purpose VLMs and specialized OCR systems at comparable cost. Chandra OCR performs similarly (83.1%) but at 24× the computational cost ($4,000 vs. $165). Among similarly priced systems, OLMOCR exceeds DeepSeek-OCR by 6.6 points (82.3% vs. 75.7%). Full category-level results are provided in Appendix D.

	Score	Cost	Wts	Data	Train
Qwen 2 VL 7B	61.3	$314	✓	✗	✗
Qwen 3 VL 8B	61.4	$234	✓	✗	✗
MinerU `1.3`	61.5	$595	✓	✗	✗
Gemini 2.5 Flash	62.1	$1,042	✗	✗	✗
Qwen 2.5 VL 7B	64.5	$165	✓	✗	✗
Nanonets OCR S	64.5	—	✓	✗	✗
Gemini Flash 2	66.3	$342	✗	✗	✗
Nanonets OCR2 3B	69.5	$702*	✓	✗	✗
GPT-4o	69.7	$7,951	✗	✗	✗
Marker `1.7.5`	70.1	$1,492	✓	✗	✗
GPT-4.1	71.0	$6,112	✗	✗	✗
Mistral OCR API	72.0	$1,000	✗	✗	✗
MinerU `2.5`	75.2*	$182*	✓	✗	✗
DeepSeek-OCR	75.7	$167*	✓	✗	✗
MonkeyOCR 3B	75.8*	—	✓	✗	✗
Marker `1.10`	76.1	$1,492*	✓	✗	✗
Infinity-Parser 7B	79.1*	—	✓	✓	✗
dots.OCR	79.1*	—	✓	✗	✓
PaddleOCR-VL	80.0*	$315*	✓	✗	✓
Chandra `0.1`	83.1*	$4,000*	✓	✗	✗
OLMOCR v1 0225	68.2	$176	✓	✓	✓
OLMOCR v1 0825	78.5	$165	✓	✓	✓
OLMOCR v2 1125	**82.3**	$165	✓	✓	✓

Table 1: Comparison of OCR systems on OLMOCR-BENCH. Score = OLMOCR-BENCH accuracy; Cost = estimated USD to process 10,000 pages; Wts = model weights and inference code; Data = training data; Train = training code. * = reported by authors.

5 Deployment at Scale

OLMOCR is released as a production-ready CLI tool. Quantized to FP8 and served with vLLM (Kwon et al., 2023), it processes approximately 3,500 tokens per second on a single H100 GPU, or about $165 per 10K pages. For pretraining-scale workloads, workers coordinate via lock files on a shared filesystem, allowing hundreds of instances to process a common document pool without duplication. Using this setup, we processed over 100M PDFs, totaling roughly 1T tokens, to construct pretraining data for OLMo 3 (Olmo et al., 2025), demonstrating OLMOCR's reliability at web scale.

6 Lessons from Open Development

Engineering iteration can rival modeling improvements. Many of the largest quality gains came not from retraining models, but from engineering fixes surfaced through real-world deployment and community feedback. Switching the output format from *JSON to YAML* reduced page-level retry rates from roughly 40% to around 1%, because the model no longer needed to manage

nested quoting and could simply emit an EOS token. *Dynamic temperature scaling*, which starts at 0.1 for quality and retries failed pages with increasing temperature up to 0.8, improved overall score by approximately 0.3 absolute points over fixed-temperature decoding while reducing failure rates to near zero, about 0.01%. Other practical adjustments, including automatic *rotation correction*, matching prompt element order between training and inference, and tuning *image resolution* to 1288px on the longest edge, further improved robustness without retraining. These engineering refinements account for much of the 10.3-point improvement between OLMOCR v1 0225 (68.2%) and v1 0825 (78.5%). These issues were identified by users running OLMOCR at scale, and resulting fixes were fast and easy to implement.

SFT and RLVR address different failure modes. Supervised fine-tuning (SFT) and reinforcement learning with verifiable rewards (RLVR) play complementary roles in OCR quality, as shown in Figure 7. SFT on `olmOCR-mix-1025` delivers the largest gains on layout and formatting challenges such as multi-column layouts, headers and footers, long tiny text, and degraded scans. RLVR primarily improves structured semantic categories such as math formulas and complex tables, where binary unit-test rewards directly optimize fine-grained structural correctness. The overall 5.3-point gain from RLVR comes largely from these categories.

The degraded subsets serve as a generalization test. On Old Scans, SFT provides substantial gains while RLVR adds little, suggesting that heavily degraded text is mainly a recognition problem. In contrast, Old Math Scans benefits strongly from RLVR, consistent with our finding that RL improves formula-level structural fidelity even in scanned documents. Overall, SFT improves document robustness, while RLVR sharpens structured semantics.

Binary unit tests enable iterative, targeted improvement. Binary visual unit tests provide clear, interpretable pass or fail signals that are easy to extend as new failure modes emerge. Unlike continuous distance metrics, they make it straightforward to diagnose regressions, add new coverage, and directly optimize against evaluation criteria. Their dual role as both benchmark, OLMOCR-BENCH, and RL reward creates a tight feedback loop between evaluation and training. Combined with synthetic HTML generation, this framework allows structured supervision to scale cheaply and enables targeted optimization for specific issues such as rowspan and colspan tables discovered during deployment.

7 Related Work

Traditional OCR systems rely on multi-stage pipelines combining layout detection, text recognition, and structure parsing (Mori et al., 1992; Smith, 2013). Modern open-source toolkits such as MinerU (Wang et al., 2024), Marker (Paruchuri, 2025b), and PP-OCRv5 (Cui et al., 2025) extend this paradigm by composing specialized models for document layout, table extraction, and formula recognition.

More recently, vision-language models (VLMs) have enabled end-to-end OCR by directly generating structured text from page images. Early systems such as Nougat (Blecher et al., 2023) and GOT-OCR (Wei et al., 2024) demonstrated this approach for scientific documents, while proprietary models such as GPT-4o (OpenAI et al., 2024) showed strong general document understanding. OLMOCR v1 (Poznanski et al., 2025a) distilled such capabilities into a small open 7B VLM via supervised fine-tuning.

Several works explore reinforcement learning to further improve VLM-based OCR. Infinity Parser (Wang et al., 2025a) applies GRPO with edit-distance rewards on synthetic HTML; DianJin-OCR-R1 (Chen et al., 2025) incorporates reasoning-oriented RL; and related work uses RL signals for document QA (He et al., 2025; Xiong et al., 2025). In contrast, OLMOCR v2 uses binary visual unit tests as reward signals, aligning optimization directly with structured correctness rather than continuous distance metrics.

8 Conclusion

We presented OLMOCR, a fully open OCR system built through iterative release and community feedback. From v1, which showed that a 7B VLM trained with SFT can match proprietary OCR systems, to v2, which added RLVR with visual unit tests to address structured visual-textual failures, we made steady improvements through tight coupling between evaluation and training. By releasing models, data, benchmarks, and production-grade infrastructure, we established an open feedback loop built on real-world adoption.

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 [cs.CV]*.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. 2023. Nougat: Neural optical understanding for academic documents. *Preprint*, arXiv:2308.13418.

Qian Chen, Xianyin Zhang, Lifan Guo, Feng Chen, and Chi Zhang. 2025. Dianjin-ocr-r1: Enhancing ocr capabilities via a reasoning-and-tool interleaved vision-language model. *Preprint*, arXiv:2508.13238.

Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, Yue Zhang, Wenyu Lv, Kui Huang, Yichao Zhang, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. 2025. Paddleocr 3.0 technical report. *Preprint*, arXiv:2507.05595.

DeepSeek-AI. 2025. Deepseek-ocr: Contexts optical compression. Model available at: `https://huggingface.co/deepseek-ai/DeepSeek-OCR`.

Patrick Emond. 2025. Lingua-py: Natural language detection for python. Accessed: 2025-01-06.

Zhentao He, Can Zhang, Ziheng Wu, Zhenghao Chen, Yufei Zhan, Yifan Li, Zhao Zhang, Xian Wang, and Minghui Qiu. 2025. Seeing is believing? mitigating ocr hallucinations in multimodal large language models. *Preprint*, arXiv:2506.20168.

Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. 2021. Ocr-free document understanding transformer. In *European Conference on Computer Vision*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, and 40 others. 2024. DataComp-LM: In search of the next generation of training sets for language models. *arXiv [cs.LG]*.

Kyle Lo, Joseph Chee Chang, Andrew Head, Jonathan Bragg, Amy X. Zhang, Cassidy Trier, Chloe Anastasiades, Tal August, Russell Authur, Danielle Bragg, Erin Bransom, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Yen-Sung Chen, Evie Yu-Yen Cheng, Yvonne Chou, Doug Downey, Rob Evans, and 36 others. 2024. The semantic reader project. *Commun. ACM*, 67(10):50–61.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Souvik Mandal, Ashish Talewar, Siddhant Thakuria, Paras Ahuja, and Prathamesh Juvatkar. 2025. Nanonets-ocr2: A model for transforming documents into structured markdown with intelligent content recognition and semantic tagging.

S Mori, C Y Suen, and K Yamamoto. 1992. Historical review of OCR research and development. *Proceedings of the IEEE. Institute of Electrical and Electronics Engineers*, 80(7):1029–1058.

Junbo Niu, Zheng Liu, Zhuangcheng Gu, Bin Wang, Linke Ouyang, Zhiyuan Zhao, Tao Chu, Tianyao He, Fan Wu, Qintong Zhang, Zhenjiang Jin, Guang Liang, Rui Zhang, Wenzheng Zhang, Yuan Qu, Zhifei Ren, Yuefeng Sun, Yuanhong Zheng, Dongsheng Ma, and 42 others. 2025a. Mineru2.5: A decoupled vision-language model for efficient high-resolution document parsing. *Preprint*, arXiv:2509.22186.

Junbo Niu, Zheng Liu, Zhuangcheng Gu, Bin Wang, Linke Ouyang, Zhiyuan Zhao, Tao Chu, Tianyao He, Fan Wu, Qintong Zhang, and 1 others. 2025b. Mineru2. 5: A decoupled vision-language model for efficient high-resolution document parsing. *arXiv preprint arXiv:2509.22186*.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, and 49 others. 2025. Olmo 3. *ArXiv*, 2512.13961.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, and 21 others. 2024. 2 OLMo 2 Furious. *arXiv preprint*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. 2024. Omnidocbench: Benchmarking diverse pdf document parsing with comprehensive annotations. *Preprint*, arXiv:2412.07626.

Vik Paruchuri. 2025a. chandra: OCR model that handles complex tables, forms, handwriting with full layout. https://github.com/datalab-to/chandra. GitHub repository.

Vik Paruchuri. 2025b. Marker: Convert pdf to markdown + json quickly with high accuracy. Version 1.4.0.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra-Aimée Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only. *ArXiv*, abs/2306.01116.

Jake Poznanski, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Aman Rangapur, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. 2025a. olmOCR: Unlocking Trillions of Tokens in PDFs with Vision Language Models. *Preprint*, arXiv:2502.18443.

Jake Poznanski, Luca Soldaini, and Kyle Lo. 2025b. olmOCR 2: Unit Test Rewards for Document OCR. Technical report, Allen Institute for AI.

RedNote HiLab. 2025. dots.ocr: Multilingual document layout parsing in a single vision-language model. Version 1.0, Released July 30, 2025.

Ray W Smith. 2013. History of the tesseract OCR engine: what worked and what didn't. In *Document Recognition and Retrieval XX*, volume 8658, page 865802. SPIE.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Baode Wang, Biao Wu, Weizhen Li, Meng Fang, Yanjie Liang, Zuming Huang, Haozhe Wang, Jun Huang, Ling Chen, Wei Chu, and Yuan Qi. 2025a. Infinity parser: Layout aware reinforcement learning for scanned document parsing. *Preprint*, arXiv:2506.03197.

Bin Wang, Fan Wu, Linke Ouyang, Zhuangcheng Gu, Rui Zhang, Renqiu Xia, Bo Zhang, and Conghui He. 2025b. Image over text: Transforming formula recognition evaluation with character detection matching. *Preprint*, arXiv:2409.03643.

Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, Bo Zhang, Liqun Wei, Zhihao Sui, Wei Li, Botian Shi, Yu Qiao, Dahua Lin, and Conghui He. 2024. Mineru: An open-source solution for precise document content extraction. *Preprint*, arXiv:2409.18839.

Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jianjian Sun, Yuang Peng, and 1 others. 2024. General ocr theory: Towards ocr-2.0 via a unified end-to-end model. *arXiv preprint arXiv:2409.01704*.

Junyu Xiong, Yonghui Wang, Weichao Zhao, Chenyu Liu, Bing Yin, Wengang Zhou, and Houqiang Li. 2025. Docr1: Evidence page-guided grpo for multi-page document understanding. *Preprint*, arXiv:2508.07313.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zhiyuan Zhao, Hengrui Kang, Bin Wang, and Conghui He. 2024. Doclayout-yolo: Enhancing document layout analysis through diverse synthetic data and global-to-local adaptive perception. *arXiv preprint arXiv:2410.12628*.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. Group sequence policy optimization. *Preprint*, arXiv:2507.18071.

A Benchmark Document Categories

We describe the seven document categories in OLMOCR-BENCH, including sourcing and test generation.

- **Headers Footers (HF)** Academic PDFs (§3.1) processed with DocLayout-YOLO (Zhao et al., 2024) to detect non-content (`abandon`) regions. After masking content, a proprietary VLM extracts text from unmasked regions. We define absence tests within the first/last N characters (e.g., ensuring a page number "5" does not appear in the last 20 characters).
- **Old Scans (OS)** Historical letters and typewritten documents from the Library of Congress with human transcriptions. We create text presence/absence and reading order tests, all manually reviewed.
- **Multi Column (MC)** Multi-column academic PDFs identified visually. Claude Sonnet 3.7 renders pages to HTML to extract ordered text segments, which are manually verified. Tests use simple coherent text blocks and exclude math, superscripts, and subscripts.
- **Long Tiny Text (LTT)** Dense small-print documents (e.g., dictionaries) from the Internet Archive. GPT-4o extracts passages for text presence tests, followed by manual verification.
- **arXiv Math (AM)** Math papers with paired LaTeX and PDFs. We align OCR output with source formulas, ensure KaTeX compatibility, remove custom macros, split compound equations, and manually review.
- **Old Math Scans (OMS)** Public domain textbooks from the Internet Archive. Formula-heavy pages are selected and equations manually annotated as ground truth.
- **Tables (TA)** Academic PDFs filtered from our repository (§3.1). Gemini-Flash-2.0 generates cell relationship tests from sampled tables, which are manually verified, including complex rowspan/colspan cases.

B SFT Data and Training Details

Data filtering pipeline. We apply Lingua (Emond, 2025) to retain only English documents and remove PDFs that fail `pypdf` parsing, contain spam keywords, contain fillable forms, or have insufficient text. We globally deduplicate documents using the source URL and remove any documents that were used for OLMOCR-BENCH. We sample up to three pages uniformly at random from each qualifying PDF.

OCR target filtering. For supervision, we generate OCR output from GPT-4.1 and filter low-quality output. We exclude pages when LaTeX formulas or HTML tables failed to properly render, or when output was Markdown (which does not support rowspan/colspan) instead of HTML, among others.

Training hyperparameters. Training takes 22 hours on a single B200 GPU. We take advantage of the 192GB of GPU RAM to avoid the complexity of any FSDP sharding, and run a full finetune with batch size of one and gradient accumulation of 32. We cap the maximum length of each sample to 8,192 tokens, and use a learning rate of $2e^{-5}$ with a linear decay schedule with 10% warmup. We use AdamW (Loshchilov and Hutter, 2019) with weight decay 0.01 and a gradient norm cap of 1.0.

C RLVR Training Details

GRPO enables efficient on-policy learning by scoring multiple completions per input and using relative performance within each group to compute policy gradients, avoiding the need for a separate value model. For each page in `olmOCR2-synthmix-1025`, we generate 28 completions and score each using the proportion of visual unit tests passed (between 0.0 and 1.0).

We train for one epoch on `olmOCR2-synthmix-1025` using an 8×H100 GPU node. We use the TRL library (von Werra et al., 2020) with KL divergence coefficient $\beta = 0.01$ to prevent excessive drift from the SFT model. We soup our released model out of six random seeds.

D Full Benchmark Results

Table 2 provides the full category-level breakdown.

E Pricing Calculations

Costs were computed using publicly listed API pricing (Nov 2025): GPT-4o at \$1.25/\$5.00 per million input/output tokens, GPT-4.1 at \$1.00/\$4.00, Gemini Flash 2.0 at \$0.05/\$0.20, Gemini Flash 2.5 at \$0.15/\$1.25, Mistral OCR API at \$1 per 1,000 pages, and Chandra OCR

Model	Cost	Overall	*Layout / Formatting*				*Structured Semantic*		*Degraded or Scanned*	
			Base	Multi column	Long tiny text	Headers & footers	Arxiv Math	Tables	Old scans	Old Math Scans
Prompting General VLMs										
Gemini-2.0-Flash	$342	66.3 ± 1.2	99.1	63.5	80.3	35.0	73.9	75.6	31.0	71.6
Gemini-2.5-Flash	$1,042	62.1 ± 1.1	82.1	54.8	34.4	85.9	66.6	61.6	42.2	69.2
GPT-4o	$7,951	69.7 ± 1.1	97.6	71.9	56.8	92.0	53.1	70.4	40.5	75.3
GPT-4.1	$6,112	71.0 ± 1.2	99.0	72.5	60.2	93.6	53.7	72.7	41.4	74.9
Qwen2-VL-7B-Instruct	$314	61.3 ± 1.1	87.4	60.6	48.6	75.0	63.0	56.7	34.0	64.8
Qwen2.5-VL-7B-Instruct	$165	64.5 ± 1.1	96.5	62.8	46.8	74.5	67.9	59.2	37.8	70.1
Qwen3-VL-8B-Instruct	$234	61.4 ± 1.0	93.0	64.8	38.2	64.3	70.1	49.6	35.0	75.8
OCR tools or OCR-specialized VLMs										
Mistral OCR API	$1,000	72.0 ± 1.1	99.4	71.3	77.1	93.6	77.2	60.6	29.3	67.5
MinerU 1.3.10 (Wang et al., 2024)	$595	61.5 ± 1.1	96.6	59.0	39.1	96.6	75.4	60.9	17.3	47.4
dots.OCR* (RedNote HiLab, 2025)	—	79.1 ± 1.0	99.5	82.4	81.2	94.1	82.1	88.3	40.9	64.2
Infinity-Parser 7B* (Wang et al., 2025a)	—	$79.1 \pm ?$	98.9	82.9	84.6	92.5	87.2	69.3	48.5	69.0
Marker 1.7.5 (Paruchuri, 2025b)	$1,492	70.1 ± 1.1	99.1	72.9	84.6	84.9	76.0	57.6	27.8	57.9
Marker 1.10.1 (Paruchuri, 2025b)	$1,492*	76.1 ± 1.1	99.3	80.0	85.7	86.6	83.8	72.9	33.5	66.8
MinerU 2.5.4* (Niu et al., 2025b)	$182*	75.2 ± 1.1	93.7	78.2	83.5	96.6	76.6	84.9	33.7	54.6
PaddleOCR-VL* (Cui et al., 2025)	$315*	80.0 ± 1.0	98.5	79.9	85.7	97.0	85.7	84.1	37.8	71.0
Nanonets-OCR2-3B (Mandal et al., 2025)	$702*	69.5 ± 1.1	99.6	81.9	93.0	32.1	75.4	86.8	40.9	46.1
DeepSeek-OCR* (DeepSeek-AI, 2025)	$167*	$75.7 \pm ?$	99.8	66.4	79.4	96.1	77.2	80.2	33.3	73.6
Chandra OCR 0.1.0* (Paruchuri, 2025a)	$4,000*	83.1 ± 0.9	99.9	81.2	92.3	90.8	82.2	88.0	50.4	80.3
OLMOCR v1 (Feb 2025 original release)	$176	68.2 ± 1.1	97.9	67.6	54.8	93.4	63.3	62.3	38.6	67.5
OLMOCR v1 (August 2025 update)	$165	78.5 ± 1.1	98.9	77.3	81.2	95.1	78.6	72.9	43.9	79.9
OLMOCR v2 (Nov 2025 release)	$165	82.3 ± 0.9	99.9	83.1	83.0	95.8	83.3	85.2	47.5	80.8

Table 2: Our full recipe vs baseline performance on OLMOCR-BENCH. Results are fully reproduced by ourselves, except those marked with * which are reported by their authors. In cases where our reproduction was lower than reported, we use the reported scores. Pricing breakdown based on API costs or hourly GPU rates, which we detail in Appendix E.

API at $4 per 1,000 pages. For locally run models, we use GPU on-demand rates from RunPod (Oct 2025): L40S $0.79/hr, A100 $1.39/hr, H100 $2.69/hr. MinerU 1.3.10 processed 1,288 pages in 58 minutes on an L40S; Marker v1.7.5 processed 10K pages in 5h31m on an H100 (with `force_ocr`); OLMOCR processed 10K pages in 36m47s on a single H100 with FP8 quantization. OLMOCR-7B and Qwen2.5VL use the same setup and cost assumptions; Qwen3VL cost is scaled by measured throughput ratio (2,100 vs 3,100 tok/s on H100). When local replication was infeasible, we report benchmark and performance numbers from the original papers: MinerU 2.5.4 (Niu et al., 2025a), PaddleOCR-VL (Cui et al., 2025), Nanonets-OCR2-3B (Niu et al., 2025a), DeepSeek-OCR (DeepSeek-AI, 2025). All reported costs reflect batch-mode pricing where available.