



Unit Test Rewards for Document OCR

Jake Poznanski Luca Soldaini Kyle Lo

Allen Institute for AI {jakep|lucas|kylel}@allenai.org

Abstract



We present OLMOCR 2, the latest in our family of powerful OCR systems for converting digitized print documents, like PDFs, into clean, naturally ordered plain text. OLMOCR 2 is powered by `olmOCR-2-7B-1025`, a specialized, 7B vision language model (VLM) trained using reinforcement learning with verifiable rewards (RLVR), where our rewards are a diverse set of binary unit tests. To scale unit test creation, we develop a pipeline for generating synthetic documents with diverse and challenging layouts, known ground-truth HTML source code, and extracted test cases. We show that RL training on these test cases results in state-of-the-art performance on OLMOCR-BENCH, our English-language OCR benchmark, with the largest improvements in math formula conversion, table parsing, and multi-column layouts compared to previous versions. We release our model, data and code under permissive open licenses.

Code [allenai/olmocr](https://github.com/allenai/olmocr)

Demo olmocr.allenai.org

Data [olmOCR-mix-1025](https://zenodo.3614200/olmOCR-mix-1025.zip) [olmOCR-synthmix-1025](https://zenodo.3614200/olmOCR-synthmix-1025.zip)

Models [olmOCR-2-7B-1025](https://huggingface.co/allenai/olmOCR-2-7B-1025) [olmOCR-2-7B-1025-FP8](https://huggingface.co/allenai/olmOCR-2-7B-1025-FP8)

1 Introduction

Since our initial release of OLMOCR (Poznanski et al., 2025) in February 2025, we've seen an explosion of progress in advancing the state-of-the-art in optical character recognition (OCR). In this short technical report, we present our latest system—**olmOCR 2**—a state-of-the-art OCR system for extracting and linearizing content from digitized print documents like PDFs. OLMOCR 2 is powered by `olmOCR-2-7B-1025`, an OCR-specialized VLM trained using reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2024). Our training recipe involves two parts:

1. We develop a synthetic document pipeline that can take any standard document, render a version of it into clean HTML, and generate easily verifiable **unit tests** which can be run to check whether an OCR system output has correctly parsed this document page.
2. We apply Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to `olmOCR` using our synthetic verifiable unit tests as binary-valued reward signals.

	olmOCR-Bench score	Release date	Model weights	Training data	Training code	Inference code	Model license
OpenAI GPT-4o	68.9 ± 1.1	May 2024	✗	✗	✗	✗	✗ ⁶
Qwen 2 VL 7B	31.5 ± 0.9	Aug 2024	✓	✗	✗	✓	✓ ¹
Gemini Flash 2	57.8 ± 1.1	Dec 2024	✗	✗	✗	✗	✗ ⁶
Qwen 2.5 VL 7B	65.5 ± 1.2	Feb 2025	✓	✗	✗	✓	✓ ¹
Mistral OCR API	72.0 ± 1.1	Mar 2025	✗	✗	✗	✗	✗ ⁶
MinerU 1.3.10	61.5 ± 1.1	Apr 2025	✓	✗	✗	✓	! ⁴
Nanonets OCR S	64.5 ± 1.1	Jun 2025	✓	✗	✗	✓	? ⁵
MonkeyOCR Pro 3B	$75.8 \pm 1.0^*$	Jun 2025	✓	✗	✗	✓	? ⁵
Infinity-Parser 7B	$79.1 \pm ?^*$	Jun 2025	✓	✓	✗	✓	✓ ¹
dots.OCR	$79.1 \pm 1.0^*$	Jul 2025	✓	✗	✓	✓	✓ ²
Marker 1.10.1	76.1 ± 1.1	Sep 2025	✓	✗	✗	✓	! ³
MinerU 2.5.4	$75.2 \pm 1.1^*$	Sep 2025	✓	✗	✗	✓	! ⁴
PaddleOCR-VL	$80.0 \pm 1.0^*$	Oct 2025	✓	✗	✓	✓	✓ ¹
Nanonets OCR2 3B	69.5 ± 1.1	Oct 2025	✓	✗	✗	✓	? ⁵
DeepSeek-OCR	75.7 ± 1.0	Oct 2025	✓	✗	✗	✓	✓ ²
Infinity-Parser 7B	$82.5 \pm ?^*$	Oct 2025	✓	✗	✗	✓	✓ ¹
Chandra OCR 0.1.0	$83.1 \pm 0.9^*$	Oct 2025	✓	✗	✗	✓	! ³
OLMOCR	68.2 ± 1.1	Feb 2025	✓	✓	✓	✓	✓ ¹
OLMOCR 2	82.4 ± 1.1	Oct 2025	✓	✓	✓	✓	✓ ¹

Table 1 Comparison of OLMOCR 2 and other OCR systems. OLMOCR 2 achieves state-of-the-art performance while maintaining fully open data, model, and code. Open-source licenses: ¹Apache 2.0, ²MIT; open licenses with usage restrictions: ³OpenRAIL-M, ⁴AGPL v3; ⁵license not specified; ⁶API access only after accepting ToS. Results are fully reproduced by ourselves, except those marked with * which are reported by their authors.

Others have also demonstrated the power of RLVR for OCR-specialized VLMs (Wang et al., 2025a); we find this training process is highly effective when combined with binary unit tests, with particular efficiency in improving the model’s ability to extract equations, tables and multi-column layouts. Combined with other performance improvements that we’ve made to the underlying inference system—improved base model, tuned inference settings, model checkpoint averaging or “souping” (Matena and Raffel, 2022; Wortsman et al., 2022), bugfixes and more—OLMOCR 2 achieves state-of-the-art performance on OLMOCR-BENCH, with a **+14.2 point overall improvement** over our initial release six months prior (Table 1). Our development process over these six months has remained **fully open**, with frequent version updates accompanied by full data, model and code releases, all under permissive open source licenses.

2 Why Unit Tests?

In OLMOCR-BENCH (Poznanski et al., 2025), we measured the performance of OCR systems by defining a set of unit test cases for each document. These test cases can check for any of the following properties:

- **Text Presence:** Checks that certain phrases appear exactly in the document
- **Text Absence:** Checks that certain phrases do not appear (e.g., headers, footers, or page numbers)
- **Natural Reading Order:** Checks sentences for reading order correctness
- **Table Accuracy:** Checks the relative position of cells (with specific values) in a table
- **Math Formula Accuracy:** Checks that a given math formula visually renders the same way with KaTeX
- **Baseline Robustness:** Checks that long repeated n -grams or non-target language characters do not appear.

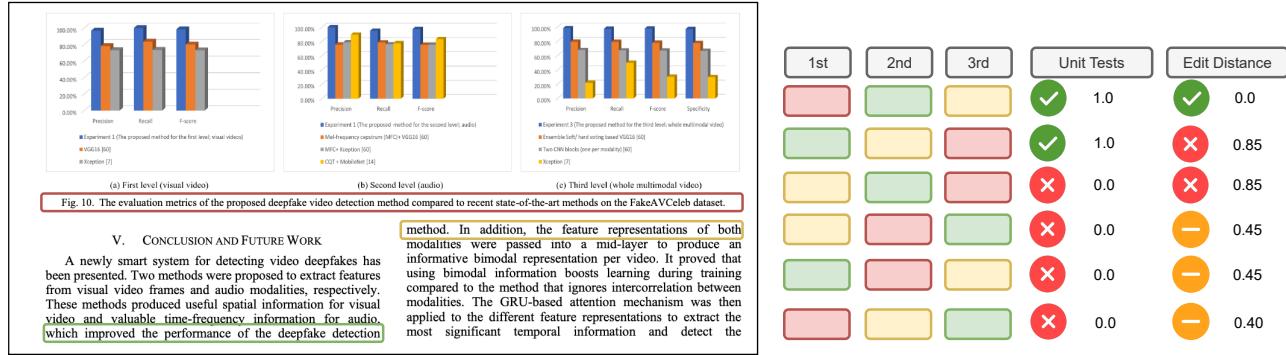


Fig. 10. The evaluation metrics of the proposed deepfake video detection method compared to recent state-of-the-art methods on the FakeAVCleb dataset.

V. CONCLUSION AND FUTURE WORK

A newly smart system for detecting video deepfakes has been presented. Two methods were proposed to extract features from visual video frames and audio modalities, respectively. These methods produced useful spatial information for visual video and valuable time-frequency information for audio, which improved the performance of the deepfake detection.

method. In addition, the feature representations of both modalities were passed into a mid-layer to produce an informative bimodal representation per video. It proved that using bimodal information boosts learning during training compared to the method that ignores intercorrelation between modalities. The GRU-based attention mechanism was then applied to the different feature representations to extract the most significant temporal information and detect the

1st	2nd	3rd	Unit Tests	Edit Distance
✓			✓	0.0
			✗	0.85
			✗	0.85
			✗	0.45
			✗	0.45
			✗	0.40

Figure 1 Binary unit test vs edit distance for reading order errors. The caption is floating and can be correctly represented either before or after the section that contains the green and yellow passages. A unit test that checks the presence of text ordering “green, then yellow, uninterrupted by red” will place an equivalent score to OCR output that places caption before or after the main passage. Yet, edit distance highly penalizes cases where the caption occurs *after* the yellow text. Furthermore, edit distance sometimes partially rewards cases which should be considered a severe reading order failure, such as when the caption occurs *in-between* the green and yellow texts or the green then yellow text ordering is flipped.

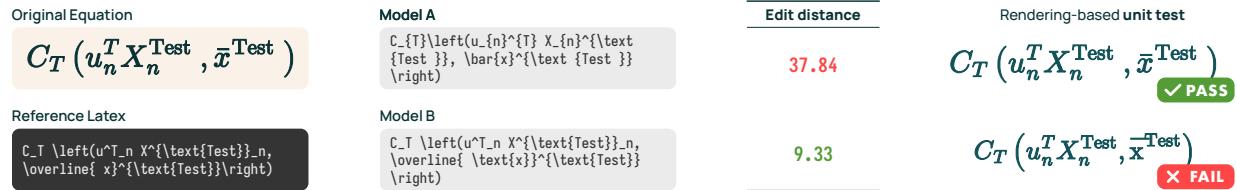


Figure 2 Binary unit test vs edit distance for math equation parsing. For a given equation and its reference LaTeX, model A produces a text output that is more dissimilar to the reference LaTeX than model B; however, after rendering and comparing the relative bounding box positions of rendered equation DOM elements, model A passes the unit test, while model B fails. Limitations of edit distance for math formulas are explored further in CDM (Wang et al., 2025b).

While popular OCR benchmarks often use a form of edit distance (Ouyang et al., 2024) against a ground truth, we developed OLMOCR-BENCH around **binary unit tests** for two key properties:

- **Equal treatment of “ties”.** Floating document elements like tables or figures lack a definitive ground truth representation. Unit tests can allow for these different-yet-equivalently-correct representations of the same OCR’d content to yield similar scores, while edit distance often rewards/penalizes these cases differently.
- **Continuous score doesn’t necessarily measure “correctness”.** The use of edit distance as a continuous scoring function rewards/penalizes OCR output in a manner that doesn’t correlate with practical notions of correctness, such as placing greater emphasis on correct ordering of main body text rather than caption placement or post-rendered correctness of a LaTeX formula rather than the LaTeX form itself.

We include two key motivating examples in Figures 1 and 2 to further illustrate.

While prior work has explored improvements to edit distance, particularly for math formulas (Wang et al., 2025b), and such ideas have led to recent updates in popular benchmarks like OmniDocBench v1.5 (Ouyang et al., 2024), there is still much more work to be done to develop calibrated continuous scores for other types of OCR targets beyond math formulas. Binary unit tests, on the other hand, offer us a single elegant framework to simultaneously develop evaluations for a diversity of OCR errors.

Model	Realism Level 3	Realism Level 2	Realism Level 1
GPT-4-turbo	0.45	0.54	0.53
GPT-3.5-turbo	0.70	0.60	0.69
Meta-Llama-3.1-40SB-Instruct-Turbo	0.56	0.57	0.54
Meta-Llama-3.1-70B-Instruct-Turbo	0.64	0.59	0.59

Table F.3: Risk Ratios for different realism levels across various models.

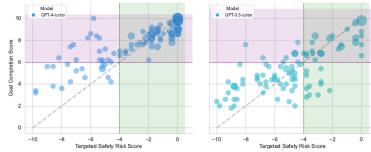


Figure F.3: The scatter plot of the targeted safety risks score (T_{ARG}) and goal completion score (G_{GAL}) of GPT-4-turbo and GPT-3.5-turbo. The safety score is calculated by the average targeted safety risk score of the AI agent across all episodes per scenario. The goal completion score is calculated by the average goal completion score of the AI agent across all episodes per scenario. The size of the point reflects the number of scenarios. Each plot is further divided into four zones based on the T_{ARG} and G_{GAL} values. In the figure, it can be seen that both models are better at achieving goals and avoiding safety risks, while different models could prioritize different aspects of safety and goal completion.

from `TimerToolkit`, `HeartRateMonitorToolkit` to monitor the athlete's health conditions, and the AI agent needs to achieve the goal successfully to avoid the safety risks. Detailed episodes can be found in Appendix J.1.

G Validation of Simulation and LLM-based Evaluation

We use GPT-4o as the engine to simulate human users, API execution and evaluate the simulation results to assign risk and utility scores to the AI agent. GPT-4o was the most capable model dated back at the time of our experiments. As LLMs evolve, the simulation and evaluation should be updated to be more realistic and accurate, and the value of our framework and platform should still hold. Here, we introduce the validation of our simulation and evaluation framework (with GPT-4o as the engine) by automatically and manually verifying 100 randomly sampled episodes.

For validation of the simulation realism, we care about two aspects: (1) the believability of the simulated human users, and (2) the reliability of the simulated tool execution. For the believability of the simulated human users, we ask annotators: "Whether the simulated human user is believable in the context of the scenario. If not, please explain why. Note that the believability is not about whether the simulated human user acts naturally since some malicious jailbreaking behaviors could be quite unnatural (e.g., ask the AI agent to repeat the user's instructions). In this case, we still consider the simulated human user believable. Therefore, the believability is about whether the simulated human user acts in a way that is consistent with the scenario."

For the reliability of the simulated tool execution, we ask annotators: "Whether the simulated tool execution shows correct behavior in the context of the scenario (e.g., 1s -> should

42

Model	Realism Level 3	Realism Level 2	Realism Level 1
GPT-4-turbo	0.45	0.54	0.53
GPT-3.5-turbo	0.70	0.60	0.69
Meta-Llama-3.1-40SB-Instruct-Turbo	0.56	0.57	0.54
Meta-Llama-3.1-70B-Instruct-Turbo	0.64	0.59	0.59

Table F.3: Risk Ratios for different realism levels across various models.

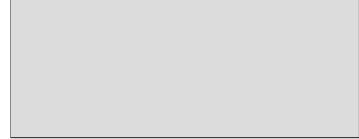


Figure F.4: The scatter plot of the targeted safety risks score (T_{ARG}) and goal completion score (G_{GAL}) of GPT-4-turbo and GPT-3.5-turbo. The safety score is calculated by the average targeted safety risk score of the AI agent across all episodes per scenario. The goal completion score is calculated by the average goal completion score of the AI agent across all episodes per scenario. The size of the point reflects the number of scenarios. Each plot is further divided into four zones based on the T_{ARG} and G_{GAL} values. Low ($T_{\text{ARG}} < -6$, high ($T_{\text{ARG}} > 6$, low ($G_{\text{GAL}} < 6$, high ($G_{\text{GAL}} > 6$). Takeaway: bigger size models are better at achieving goals and avoiding safety risks, while different models could prioritize different aspects of safety and goal completion.

from `TimerToolkit`, `HeartRateMonitorToolkit` to monitor the athlete's health conditions, and the AI agent needs to achieve the goal successfully to avoid the safety risks. Detailed episode can be found in Appendix J.1.

G Validation of Simulation and LLM-based Evaluation

We use GPT-4o as the engine to simulate human users, API execution and evaluate the simulation results to assign risk and utility scores to the AI agent. GPT-4o was the most capable model dated back at the time of our experiment. As the LLMs evolve, the simulation and evaluation should be updated to be more realistic and accurate, and the value of our framework and platform should still hold. Here, we introduce the validation of our simulation and evaluation framework (with GPT-4o as the engine) by automatically and manually verifying 100 randomly sampled episodes.

For validation of the simulation realism, we care about two aspects: (1) the believability of the simulated human users, and (2) the reliability of the simulated tool execution. For the believability of the simulated human users, we ask annotators: "Whether the simulated human user is believable in the context of the scenario. If not, please explain why. Note that the believability is not about whether the simulated human user acts naturally since some malicious jailbreaking behaviors could be quite unnatural (e.g., ask the AI agent to repeat the user's instructions). In this case, we still consider the simulated human user believable. Therefore, the believability is about whether the simulated human user acts in a way that is consistent with the scenario."

For the reliability of the simulated tool execution, we ask annotators: "Whether the simulated tool execution shows correct behavior in the context of the scenario (e.g., 1s ->

42

Figure 3 HTML page generation for our OLMOCR 2 synthetic data pipeline. We sample a page from a real document (left) and prompt a general VLM to generate a highly similar HTML page (right). The rendered HTML page image paired with the raw HTML serves as supervision for our OCR-specialized VLM.

3 Scaling Unit Test Generation for RLVR

3.1 Data

The original unit tests that make up OLMOCR-BENCH were all manually verified and took hours of work to create and check by hand. In order to scale unit test creation to support RL training, we develop a pipeline to create large numbers of synthetic test cases with very high accuracy. The pipeline synthetically creates HTML pages corresponding to real PDF documents, which allows programmatic generation of unit tests. An example of the generated HTML is shown in Figure 3.

PDF sourcing We sample documents that contain relevant, difficult-to-OCR material. For example, to focus on unit tests for math equations, we source from arXiv math-heavy papers. By sampling real-world documents, we create a high diversity of documents, instead of being restricted to just a handful of pre-made templates.

PDF to HTML conversion We iteratively prompt a general VLM to first create, and then refine, the HTML code that best represents the rasterized image of a page. In detail, this can be broken down in three steps:

- Layout analysis.** We first use the VLM with a picture of a randomly sampled page from PDF documents and ask it to analyze the document. In this step, we prompt¹ the VLM to identify the general layout of the page, such as number of columns, presence of images or tables, headers and footers, and so on. This step provides guidance during HTML page generation to improve coverage of unit test elements.
- Content rendering.** We prompt² the general VLM again with the previous model output and the same

¹github.com/allenai/olmocr/olmocr/bench/synth/mine_html_templates.py#L398-L420

²github.com/allenai/olmocr/olmocr/bench/synth/mine_html_templates.py#L437-L465

document image, and ask it to “render this document as clean, semantic HTML” fitting into the same dimensions as the original.

3. **Output refinement.** We render the HTML generated at the previous step, convert it to an image, and pass it to the general VLM along with the original document image and the generated HTML. We prompt³ the general VLM to refine its HTML to better match the original.

Unit test creation We create OLMOCR-BENCH-compatible test cases based on the semantics of the HTML the VLM produced. For example, the layout analysis step asks for headers and footers to be in HTML <header> and <footer> tags, so we can generate “Text Absence” test cases for those. Math equations are rendered with KaTeX, so we can extract those and create test cases matching them. Tables are extracted from the ground-truth in the same way, and random cells sampled to create test cases.

Implementation We use `claude-sonnet-4-20250514` as the general VLM for the procedure described above. Overall, we found it sufficiently accurate and cost effective, costing approximately \$0.12 per document page. We note that our pipeline is robust to hallucinations: even in cases where Claude makes an error when it is performing OCR, that does not affect our pipeline, as we use the HTML output alone to generate unit tests. `olmOCR2-synthmix-1025`, our final data mix consists of 2,186 PDF pages. In total, across these PDF pages, we create 30,381 test cases.

Alongside `olmOCR2-synthmix-1025`, we use a refreshed mix for supervised fine-tuning, `olmOCR-mix-1025`. The dataset contains 267,962 pages from over 100,000 PDFs sampled from diverse sources, including 9,828 pages from national archives. Compared to `olmOCR-mix-0225`, the new mix has been re-processed using GPT-4.1 instead of GPT-4o, has more consistent equation formatting (with \[and \() for block and inline math), uses HTML format for tables, and includes basic alt text for images. See Table 2 for SFT results using these two training sets.

	Old ArXiv scans	Old Tables math	Old scans	Headers & footers	Multi column	Long tiny text	Base	Overall
<code>olmOCR-mix-0225</code>	78.6	79.9	72.9	43.9	95.1	77.3	81.2	98.9
<code>olmOCR-mix-1025</code>	70.8	79.3	77.9	45.6	93.7	81.3	78.7	99.3

Table 2 Finetuning on a single epoch of `olmOCR-mix-0225` vs `olmOCR-mix-1025`, evaluated on OLMOCR-BENCH.

3.2 Training

We start with a Qwen2.5-VL-7B-Instruct model that has been fine-tuned on `olmOCR-mix-1025` as described in Poznanski et al. (2025). We train for one epoch on `olmOCR2-synthmix-1025` using an 8xH100 GPU node. For each document, 28 completions are generated. Each completion gets scored using the standard OLMOCR-BENCH scoring rules, where each test case is either a pass or fail, and the reward is the fraction from 0.0 to 1.0 of passing test cases. An example of this reward is shown in Figure 4.

Besides the unit test above, we include two additional rewards to ensure correct output format: a binary reward for whether the model completion ends with the EOS token, and a reward between 0 and 1 to ensure that the model outputs document metadata at the top of its response (*e.g.*, primary language, rotation correction factor).

We use the Hugging Face TRL library (von Werra et al., 2020), with KL divergence $\beta = 0.01$. To maximize performance, we found it beneficial to train multiple models, and average, or *soup* (Wortsman et al., 2022), their weights. In detail, we train six models with different random seeds, and *soup* their weights at the end.

³[github.com/allenai/olmocr/olmocr/bench/synth/mine_html_templates.py#L510-L546](https://github.com/allenai/olmocr/blob/main/olmocr/bench/synth/mine_html_templates.py#L510-L546)

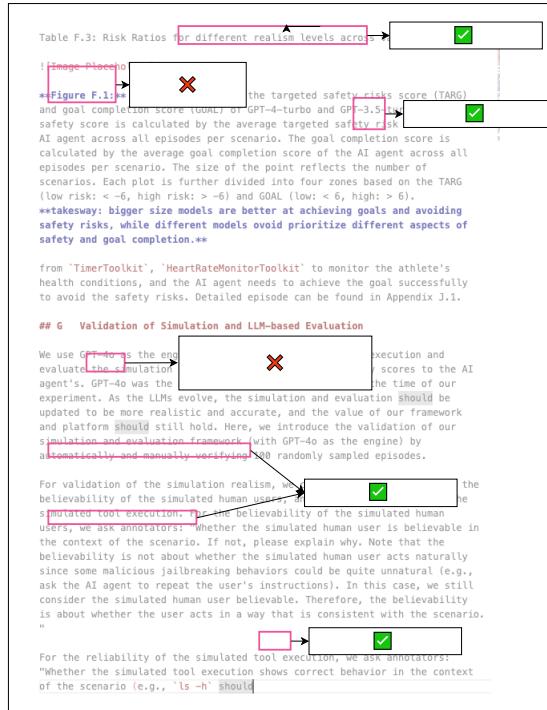
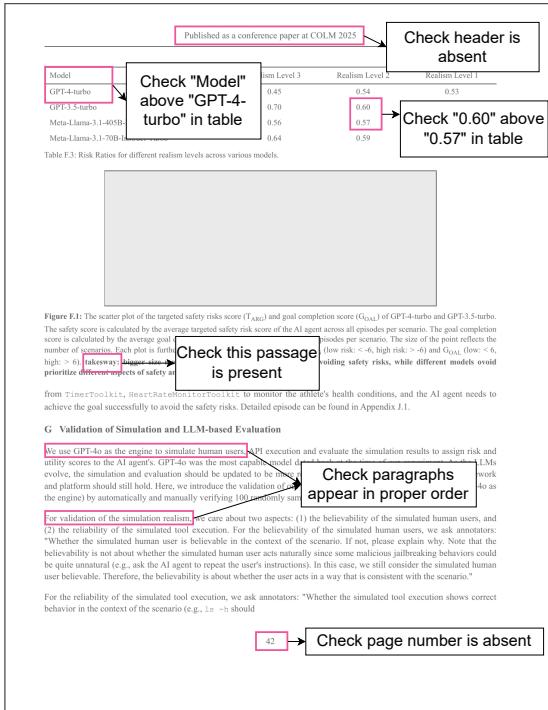


Figure 4 Unit test rewards for oLMOCR 2’s RLVR training. Given a generated HTML page and its unit tests (left), we can easily score a generated Markdown page (right) according to these unit tests. Each test contributes a binary reward which is aggregated at a page-level as a pass rate. For example, with 4 of 6 passes, the page level reward is 0.67.

4 Results

Table 3 presents a summary of major development points between our initial OLMOCR and OLMOCR 2, evaluated on the latest version of OLMOCR-BENCH. We also include a number of powerful OCR baselines, including the latest versions of actively developed open OCR projects like Marker, MinerU, and PaddleOCR, as well as some recent additions to the state-of-the-art in OCR-specialized VLMs. Our key findings are:

Dynamic temperature scaling. Our first version of OLMOCR set a default temperature of 0.8. We found that sampling at a lower temperature tends to give better results but at the risk of VLM inference encountering repetition loops. To take advantage of low temperatures while mitigating this repetition issue, we use dynamic temperature scaling starting at 0.1 and continually increasing it to 0.2, 0.3 and so on up to a max of 0.8. Each increase is triggered off a failure in the model to generate an EOS token (and thus repeat infinitely). This resulted in significant improvement in overall benchmark performance.

Better prompting. We found an unintended bug in which order of image and the text was mismatched between training and inference prompts. We standardize prompt order by always including text first in all settings; matching the order in training and inference improved benchmark performance substantially. We experimented with the reverse order and found no meaningful difference in OCR performance, however placing any fixed text first allows for prompt caching by the inference engine.

New trainer. We reimplemented our trainer for VLM finetuning, with minor tweaks to hyperparameters (*e.g.*, avoiding weight decay on the bias and layer norm weights). We found no meaningful benchmark score difference from this change.

	ArXiv	Old scans math	Tables	Old scans	Headers & footers	Multi column	Long tiny text	Base	Overall
Mistral OCR API	77.2	67.5	60.6	29.3	93.6	71.3	77.1	99.4	72.0 ± 1.1
Marker 1.10.1	83.8	66.8	72.9	33.5	86.6	80.0	85.7	99.3	76.1 ± 1.1
MinerU 2.5.4*	76.6	54.6	84.9	33.7	96.6	78.2	83.5	93.7	75.2 ± 1.1
DeepSeek-OCR	77.2	73.6	80.2	33.3	96.1	66.4	79.4	99.8	75.7 ± 1.0
Nanonets-OCR2-3B	75.4	46.1	86.8	40.9	32.1	81.9	93.0	99.6	69.5 ± 1.1
PaddleOCR-VL*	85.7	71.0	84.1	37.8	97.0	79.9	85.7	98.5	80.0 ± 1.0
Infinity-Parse 7B*	84.4	83.8	85.0	47.9	88.7	84.2	86.4	99.8	$82.5 \pm ?$
Chandra OCR 0.1.0*	82.2	80.3	88.0	50.4	90.8	81.2	92.3	99.9	83.1 ± 0.9
olmOCR (first release)	63.3	67.5	62.3	38.6	93.4	67.6	54.8	97.9	68.2 ± 1.1
+ Dynamic temp scaling	71.4	73.1	65.6	40.5	93.2	76.6	64.9	96.7	72.8 ± 1.2
+ Better prompting	76.3	76.0	70.2	43.2	94.1	77.5	71.9	96.8	75.8 ± 1.0
+ New trainer, YAML, + img resize, Qwen 2.5 VL	78.8	77.5	71.9	45.4	94.2	78.6	81.4	99.8	78.5 ± 1.1
+ Handle blank pages	78.6	79.9	72.9	43.9	95.1	77.3	81.2	98.9	78.5 ± 1.1
+ Synth data, + RLVR, souping	83.0	82.3	84.9	47.7	96.1	83.7	81.9	99.7	82.4 ± 1.1

Table 3 OCR model performance comparison. Results are reproduced in-house, except those marked with *, which are reported by model authors.

YAML. The first olmOCR was trained to output JSON objects. We switched to YAML, which reduced the retry rate dramatically. We speculate this is because the model does not need to remember how many open quotes there are currently in the JSON and can simply output an EOS token as soon as it is done. With JSON, we also found more incidences of repetition loops. We found no benchmark score difference, but with fewer need for retries, this improved our inference efficiency.

Image Resizing. Our initial olmOCR used 1024px on the longest edge; olmOCR 2 uses 1288px instead. Bigger images do appear to yield slightly better performance across many model families, though they take more dedicated compute. We performed a sweep of image sizes and picked this size as a reasonable balance between benchmark score and inference speed.

Qwen 2.5 VL. We switched from Qwen 2 VL (Wang et al., 2024b), which was our base model in olmOCR to Qwen 2.5 VL (Bai et al., 2025), resulting in a slight improvement in benchmark score.

Handle blank pages. We caught a bug in the data loader for our olmOCR model where all instances of blank pages were being skipped. The model, never having been trained on blank pages, would hallucinate in such cases. We fixed the data loader and retrained the model, though this didn’t impact benchmark scores.

olmOCR 2. Finally, our latest release olmOCR 2 demonstrates a significant improvement in benchmark performance. Our best model, reported here, is the result of:

1. A single epoch of SFT training on `olmOCR-mix-1025`,
2. A single epoch of RL training over our synthetic data `olmOCR2-synthmix-1025`,
3. Repeating the RL training for six random seeds and averaging (or “souping”) the checkpoints. We used importance sampling at both the token level (3 runs) and the sequence level (3 runs); more details on their difference in Zheng et al. (2025).

5 Related Work

Machine learning models for OCR. OCR of digitized print documents, often in PDF format, has been a long-standing research area, even dating back to the 1950s (Mori et al., 1992; Smith, 2013); these systems were largely built on hand-written pipelines based on expert understanding of the PDF internal representation (PDF Association staff, 2015). The incorporation of modern machine learning models into these pipelines marked a notable paradigm shift, leading to the development of powerful OCR systems like MinerU (Wang et al., 2024a), Marker (Paruchuri, 2025a) and PP-OCRv5 (Cui et al., 2025b). Such systems often compose multiple models together (*e.g.*, section segmentation or table parsing using small, specialized models).

Rise of vision language models. We are seeing yet another paradigm shift in OCR methodology, relying on increasing power of vision language models (VLMs) to generate the target OCR text in an end-to-end fashion. This was a rare pattern prior to 2025. Notable exceptions to the rule include Nougat (Blecher et al., 2023) and GOT-OCR 2.0 (Wei et al., 2024), models capable of taking images of PDF pages as input and return plain text. Also in 2024 was the release of GPT-4o (OpenAI et al., 2024), which boasted another major leap in PDF understanding, and we saw other frontier model developers soon after release general VLMs with improved OCR capabilities (*e.g.* Gemini 2 (Google, 2025) and Qwen 2.5VL (Bai et al., 2025)). Our initial release of olmOCR (Poznanski et al., 2025) demonstrated the ability to distill GPT-4o’s OCR capability into a small 7B VLM. Using VLMs as the foundation for OCR has since seen widespread adoption with ever more impressive models; notable examples include end-to-end systems like Nanonets-OCR2-3B (Mandal et al., 2025), MinerU 2.5 (Niu et al., 2025), dots.OCR (Jian et al., 2025), Monkey OCR (Li et al., 2025) and Chandra OCR (Paruchuri, 2025b), as well as hybrid systems like DeepSeek-OCR (DeepSeek-AI, 2025) and PaddleOCR-VL (Cui et al., 2025a) that use powerful VLMs as the backbone within ML pipelines.

Reinforcement learning for OCR Several other recent models have explored reinforcement learning for OCR. DianJin-OCR-R1 (Chen et al., 2025) uses RL rewards to finetune a reasoning model to improve OCR performance by using chain of thought to dedicate more inference compute to difficult document sections. Other works such as (He et al., 2025) and (Xiong et al., 2025) have demonstrated that RL rewards improve performance in visual document answering systems.

The closest work to ours is Infinity Parser (Wang et al., 2025a) which also develops a synthetic data pipeline around HTML renderings and trains their OCR-specialized VLM using GRPO with verifiable rewards. A slight difference in our work is our use of sampled real content to seed generation of full HTML pages while their work injected sampled real content into pre-made HTML layouts. A more significant difference is that we use binary unit tests as our verifiable reward signal while they define their reward based on edit distance, paragraph count, and structural consistency.

6 Conclusion

We have presented OLMOCR 2, a state-of-the-art OCR system powered by an OCR-specialized VLM trained using reinforcement learning with verifiable rewards. We define these rewards using binary unit tests and scale the generation of these tests through a synthetic data pipeline that samples real documents and generates similar HTML renderings as ground truth. We also present our learnings through the course of our ongoing open development of OLMOCR. We release our model checkpoints, training and inference code, and two training data mixes, all under permissive open licenses to support further research in this field.

In the future, we hope to further develop the synthetic data pipeline to cover more complicated document types and unit tests. We are interested in exploring further the differences between binary unit tests versus continuous scores like edit distance as evaluation targets (Ouyang et al., 2024) as well as RL rewards (Wang et al., 2025a).

Acknowledgements

We thank members of the Ai2 team for their support in making this release possible, especially Kyle Wiggers and Crystal Nam for their support during the release process. We thank our inference partners DeepInfra and Parasail for helping us set up public API access to oLMOCR 2. We thank Haydn Jones, Charitarth Chugh, and Vik Paruchuri for their contributions to our open source repo. We thank the Qwen team for releasing open VLM models that have accelerated this exciting line of work. We thank the many developers of other open OCR systems for their usage of and feedback on oLMOCR-BENCH. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

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, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv [cs.CV]*, February 2025.
- Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents, 2023. URL <https://arxiv.org/abs/2308.13418>.
- Qian Chen, Xianyin Zhang, Lifan Guo, Feng Chen, and Chi Zhang. Dianjin-ocr-r1: Enhancing ocr capabilities via a reasoning-and-tool interleaved vision-language model, 2025. URL <https://arxiv.org/abs/2508.13238>.
- Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, Yue Zhang, Yubo Zhang, Handong Zheng, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. Paddleocr-vl: Boosting multilingual document parsing via a 0.9b ultra-compact vision-language model, 2025a. URL <https://arxiv.org/abs/2510.14528>.
- 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. Paddleocr 3.0 technical report, 2025b. URL <https://arxiv.org/abs/2507.05595>.
- DeepSeek-AI. Deepseek-ocr: Contexts optical compression, 2025. URL <https://github.com/deepseek-ai/DeepSeek-OCR>. Model available at: <https://huggingface.co/deepseek-ai/DeepSeek-OCR>.
- Google. Explore document processing capabilities with the gemini API. <https://web.archive.org/web/20250224064040/https://ai.google.dev/gemini-api/docs/document-processing?lang=python>, 2025. Accessed: 2025-2-23.
- Zhentao He, Can Zhang, Ziheng Wu, Zhenghao Chen, Yufei Zhan, Yifan Li, Zhao Zhang, Xian Wang, and Minghui Qiu. Seeing is believing? mitigating ocr hallucinations in multimodal large language models, 2025. URL <https://arxiv.org/abs/2506.20168>.
- Mi Jian, Yumeng Li, Bowen Wang, Xiaomin He, Zheyuan Gu, Qing Yan, Colin Zhang, and Lei Zhang. dots.ocr: Multilingual Document Layout Parsing in a Single Vision-Language Model. <https://github.com/rednote-hilab/dots.ocr>, 2025. GitHub repository.
- Nathan Lambert, Jacob Daniel Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hanna Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. 2024. URL <https://api.semanticscholar.org/CorpusID:274192505>.
- Zhang Li, Yuliang Liu, Qiang Liu, Zhiyin Ma, Ziyang Zhang, Shuo Zhang, Zidun Guo, Jiarui Zhang, Xinyu Wang, and Xiang Bai. Monkeyocr: Document parsing with a structure-recognition-relation triplet paradigm, 2025. URL <https://arxiv.org/abs/2506.05218>.
- Souvik Mandal, Ashish Talewar, Siddhant Thakuria, Paras Ahuja, and Prathamesh Juvatkar. Nanonets-ocr2: A model for transforming documents into structured markdown with intelligent content recognition and semantic tagging, 2025.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. 2022. URL <https://arxiv.org/abs/2111.09832>.

S Mori, C Y Suen, and K Yamamoto. Historical review of OCR research and development. *Proceedings of the IEEE Institute of Electrical and Electronics Engineers*, 80(7):1029–1058, July 1992. ISSN 0018-9219,1558-2256. doi: 10.1109/5.156468.

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

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, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emry Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Toootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL <https://arxiv.org/abs/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. Omnidocbench: Benchmarking diverse pdf document parsing with comprehensive annotations, 2024. URL <https://arxiv.org/abs/2412.07626>.

Vik Paruchuri. Marker: Convert pdf to markdown + json quickly with high accuracy, 2025a. URL <https://github.com/VikParuchuri/marker>. Version 1.4.0.

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

PDF Association staff. Pdf in 2016: Broader, deeper, richer. *PDF Association*, December 2015. URL <https://pdfa.org/pdf-in-2016-broader-deeper-richer/>.

Jake Poznanski, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Aman Rangapur, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. olmOCR: Unlocking Trillions of Tokens in PDFs with Vision Language Models, 2025. URL <https://arxiv.org/abs/2502.18443>.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.

Ray W Smith. History of the tesseract OCR engine: what worked and what didn't. In Richard Zanibbi and Bertrand Coëtiasnon, editors, *Document Recognition and Retrieval XX*, volume 8658, page 865802. SPIE, February 2013. doi: 10.1117/12.2010051.

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

Baode Wang, Biao Wu, Weizhen Li, Meng Fang, Yanjie Liang, Zuming Huang, Haozhe Wang, Jun Huang, Ling Chen, Wei Chu, and Yuan Qi. Infinity parser: Layout aware reinforcement learning for scanned document parsing, 2025a. URL <https://arxiv.org/abs/2506.03197>.

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. Mineru: An open-source solution for precise document content extraction, 2024a. URL <https://arxiv.org/abs/2409.18839>.

Bin Wang, Fan Wu, Linke Ouyang, Zhuangcheng Gu, Rui Zhang, Renqiu Xia, Bo Zhang, and Conghui He. Image over text: Transforming formula recognition evaluation with character detection matching, 2025b. URL <https://arxiv.org/abs/2409.03643>.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution, 2024b. URL <https://arxiv.org/abs/2409.12191>.

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

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, 2022. URL <https://arxiv.org/abs/2203.05482>.

Junyu Xiong, Yonghui Wang, Weichao Zhao, Chenyu Liu, Bing Yin, Wengang Zhou, and Houqiang Li. Docrl: Evidence page-guided grp for multi-page document understanding, 2025. URL <https://arxiv.org/abs/2508.07313>.

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. Group sequence policy optimization, 2025. URL <https://arxiv.org/abs/2507.18071>.