Post-Training Kimi-K2.7 on GDP.pdf

Benchmark results and transfer to tool-using professional tasks on GDPval

Logan Ritchie* Sushant Mehta Liudas Panavas Edwin Chen

Surge AI · September 2026

Abstract

We post-trained Kimi-K2.7 on 1,465 GDP.pdf tasks, professional reasoning problems over a single PDF answered in one turn without tools, and evaluated it on GDPval, where the model must locate its input files with shell and Python tools and produce spreadsheets, documents and presentations. Although training involved no tools, the trained model worked differently on GDPval: it covered more of the named source files before building, made fewer and less repetitive read-only tool calls, consolidated its reasoning into fewer, longer steps, and checked its output more often. Paired examples from both benchmarks show the same four skills: understanding what a source represents, checking that the evidence covers the whole task, resolving inconsistencies that change the answer, and carrying every requirement through to the deliverable. Scores rose accordingly. On the held-out public GDP.pdf split, the full-task pass rate rose from 11.0% to 24.5%, moving the model from 35th to 9th of 41 entries on the public leaderboard. On GDPval, the share of tasks earning at least 90% of rubric points rose from 19.5% to 32.7%, perfect scores doubled from 6 to 13, and 119 tasks improved against 63 that regressed, with gains in sectors outside the training domains and in tasks with no PDF inputs. Training used two-stage LoRA: on-policy self-distillation against private rubric feedback, then rubric-rewarded reinforcement learning.

1 Introduction

This report studies whether reasoning behaviour learned by post-training a model on single-document tasks transfers to a tool-using, multi-file setting. We post-trained Kimi-K2.7 [Moonshot AI, 2026] on GDP.pdf [Garre et al., 2026], professional reasoning tasks based on individual PDF files. In this study, PDF contents were supplied directly as extracted text and images, and the model answers in one turn without tools. We evaluated the resulting checkpoint on the held-out public GDP.pdf split and on GDPval [Patwardhan et al., 2025], in which the model works inside a sandbox, locates its input files with shell and Python tools, and produces a deliverable such as a spreadsheet, report, presentation or PDF. The two evaluations ask for the same kind of professional work but differ in interaction mode, input format and domain coverage. Only 38 of the 220 GDPval tasks have a PDF among their inputs, and 95 have no input files.

The main findings are as follows.

1. On the public GDP.pdf split, the full-task pass rate rose from 11.0% to 24.5%, moving the model from 35th to 9th of 41 entries on the public leaderboard. The gain is present in all ten domains and does not attenuate on inputs longer than the training context.
2. On GDPval, the share of tasks earning at least 90% of rubric points rose from 19.5% to 32.7%, perfect scores rose from 6 to 13, and 119 tasks improved against 63 that regressed. Gains were present in the four sectors with no counterpart among the training domains and in tasks with no PDF among their inputs.
3. Trajectory analysis on GDPval shows that the trained model accessed more of the named source files before building a deliverable, made fewer read-only tool calls, retrieved less and less repetitive text, consolidated its reasoning into fewer and longer messages, and more often ran a check on what it had written.

*Correspondence to: research@surgehq.ai

4. Paired examples from both benchmarks show the same four skills: understanding what the source material represents, checking that it covers the whole task, recognizing when a missing detail or inconsistency changes the answer, and carrying every requirement through to the final output.

An earlier experiment[1] found that a model post-trained on office tasks changed its working behaviour on software-engineering tasks. The present experiment tests transfer across a different boundary: the professional reasoning in training and evaluation is related, but the way the model interacts with its inputs is not. Despite no tool use involved in the training tasks, the way Kimi-K2.7 leveraged available tools in the subsequent GDPval evaluation changed significantly. This indicates that the model was very able to adapt what it learned in training into a very different setting.

The report is organized as follows. Section 2 describes the training data, recipe and evaluations. Section 3 gives the benchmark results. Section 4 reports changes in tool use and reasoning on GDPval, and Section 5 gives paired trajectory examples from both benchmarks. Section 6 covers limitations and remaining failures. Training details, harness settings, and metric definitions are in the appendices.

2 Setup

Training data. We trained on 1,465 GDP.pdf tasks spanning ten professional domains: finance and investing, STEM and research, construction, healthcare, real estate, manufacturing and supply chains, HR, insurance, legal and engineering. Each task pairs a real professional PDF with a workflow question written by a domain expert about work they recently performed, and with an atomic rubric of natural-language grading criteria, about ten per task and as many as 30. The rubrics reward both the primary intent of the question and "dodged bullets": errors a professional would recognize as disqualifying even though the prompt never mentions them. The source PDFs run to 73,609 pages in total, about fifty per task, and much of their evidence sits in tables, drawings and charts rather than prose. The tasks follow the construction protocol of the public GDP.pdf benchmark and are disjoint from its 100 public evaluation tasks.

Training method. We adapted Kimi-K2.7 with rank-32 LoRA [Hu et al., 2022, Schulman and Thinking Machines Lab, 2025] in two stages, using 600 tasks for the first and 865 for the second, both balanced across the ten domains.

- *On-policy self-distillation (OPSD).* For each task a strong model wrote targeted, private feedback derived from the rubric, describing what a previous attempt missed and why, without stating the answer. The student sampled a response from the public prompt. The same model with the same adapter, prompted with the public prompt plus the private feedback, re-scored that exact token sequence, and the per-token reverse KL between the two distributions was the only training signal [Zhao et al., 2026, Lu and Thinking Machines Lab, 2025]. This pulled the model toward a version of its own response informed by the rubric. Evaluation never saw the feedback.
- *Rubric-rewarded reinforcement learning.* GSPO [Zheng et al., 2025] with group-relative advantages [Shao et al., 2024], 64 prompts and 8 samples per update, and a dense reward equal to the fraction of rubric criteria that an LLM grader, GPT-5.4 Mini at medium reasoning, judged satisfied, in the spirit of rubric-based rewards [Gunjal et al., 2026, Viswanathan et al., 2025].

Both stages ran at a 65,536-token context. Full hyperparameters and infrastructure are in Appendix A.

Training environment. Each training example was one document and one question. The document entered the prompt as its full extracted text, a short guide to likely-relevant pages, and rendered page images for up to sixteen pages where non-text content mattered. The model had no tools and answered in a single turn. Appendix B gives the input construction.

Evaluation. We evaluated the final checkpoint on two benchmarks.

[1] https://surgehq.ai/blog/office-work-post-training-improves-coding

- *GDP.pdf*: the 100 public tasks, over documents never seen in training, with the same input construction as training but served in a 262,144-token window so that inputs of up to 240K tokens are seen in full. Inputs range from 6K to 240K tokens with a median of 87K, and 72 of the 100 exceed the 64K training context. A task passes only if every rubric criterion is satisfied, as judged by an LLM against the benchmark's criterion verifiers. The base model was run once and the trained model four times.
- *GDPval*: the 220 gold tasks, written by professionals in 44 occupations across nine sectors, run through the Inspect Evals harness [UK AI Security Institute, 2024, UK AI Security Institute et al., 2024]. The model has Python and shell tools in a sandbox, must discover its input files in a `reference_files` folder, writes deliverables to a `deliverable_files` folder, and ends with a message that is graded together with the files by GPT-5.4 Mini at medium reasoning against the task rubric. Scores are the fraction of rubric points awarded. Sessions average about seventy messages. Sector, occupation and input-file metadata come from the public task release.

The aggregate scores in Section 3 are from the full 220-task GDPval evaluation. The trajectory statistics and side-by-side examples in Sections 4 and 5 come from a 100-task replication of the base-versus-trained comparison run in our own harness with artifact-native grading, in which every metric is computed as a paired difference within that run.

3 Performance gains on GDP.pdf and GDPval

Table 1 summarizes the results. Strict full-task success roughly doubled on both benchmarks, and more tasks improved than regressed.

Table 1: Headline results. GDP.pdf: 100 public tasks; the trained figures average four runs. Full-task pass requires every rubric criterion; GDPval full-task pass is a perfect rubric score. GDP.pdf direction counts tasks whose mean criteria pass rate over the four trained runs is above, below or equal to the base run; GDPval direction is the paired change in rubric score.

Evaluation	Full-task pass	Score $\geq 90\%$	Improved / regressed / tied
GDP.pdf (100 tasks)	11.0% $\rightarrow$ **24.5%**	27.0% $\rightarrow$ **36.0%**	**53** / **29** / **18** tasks
GDPval (220 tasks)	2.7% $\rightarrow$ **5.9%**	19.5% $\rightarrow$ **32.7%**	**119** / **63** / **38** tasks

3.1 GDP.pdf

Each of the four trained runs beat the base model, solving 27, 24, 24 and 23 of the 100 tasks against 11 (exact McNemar $p \leq 0.003$ for every run), and 44 tasks were solved in at least one run. Inserted into the public leaderboard[2] (Figure 1), the four-run mean of 24.5% takes 9th place, 0.2 points below GPT-5.6 Terra (Medium) and above Claude Opus 4.8 (Adaptive/Max), Claude Opus 5 (Adaptive/Max), Gemini 3.7 Flash (High), Gemini 3.8 Flash (Medium and High), Qwen 3.8 Max (xHigh) and Kimi K3 (Max). The base model's 11.0% would rank 35th.

The gain is present in all ten domains, from HR (22% to 42%) and healthcare (18% to 43%) to drawing-heavy construction and engineering. It also does not attenuate with document length: on inputs at or below the 64K training context ($n = 28$) pass rate rises from 17.9% to 32.1%, and on inputs above it ($n = 72$) from 8.3% to 21.5%, with the largest single-bucket gain on inputs above 128K tokens (Appendix C).

3.2 GDPval

GDPval differs from the training setting in every respect except the kind of work asked for: it is multi-turn, tool-using and multi-file, and it is judged on model-created artifacts, whereas training was single-turn, single-document and tool-free. Table 2 gives the results. Tasks earning at least 90% of their rubric points rise

[2] `https://surgehq.ai/benchmarks/gdp-pdf`, retrieved September 12, 2026; 41 entries.

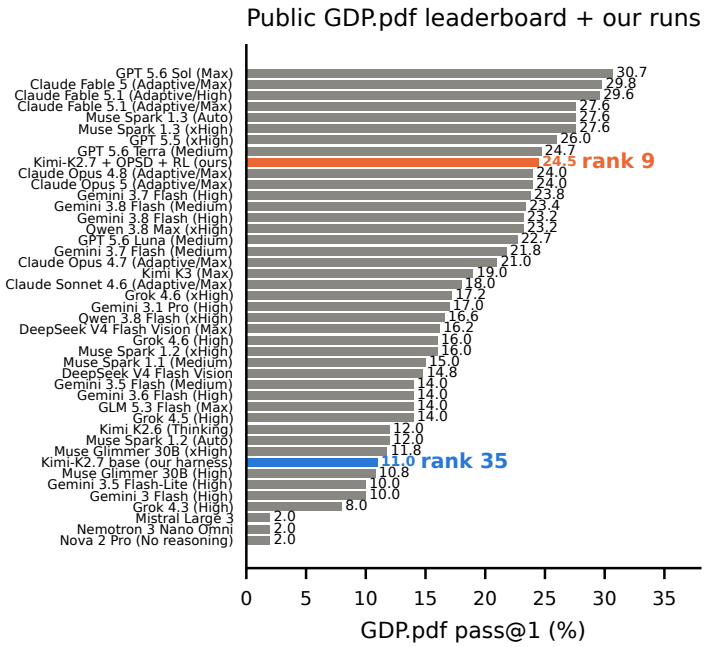

Figure 1: The public GDP.pdf leaderboard (41 entries, September 2026) with our two runs inserted. Training moves Kimi-K2.7 from 11.0% (35th) to 24.5% (9th) full-task pass rate. Our two scores come from our own harness; leaderboard entries are scored by the benchmark maintainers (Appendix B).

from 43 to 72 and perfect scores from 6 to 13. Paired per task, 119 tasks improve and 63 regress (62 against 24 by more than 0.10 of the rubric; Wilcoxon $p < 10^{-6}$), and at the 90% threshold 38 tasks move up while 9 move down (McNemar $p = 2 \times 10^{-5}$).

Table 2: GDPval: number and share of the 220 tasks whose rubric score reaches each threshold.

Model	$\geq 90\%$	100%
Base Kimi-K2.7	43 (19.5%)	6 (2.7%)
OPSD + RL	**72 (32.7%)**	**13 (5.9%)**

Sectors outside the training domains. Eight of the nine sectors improve at the 90% threshold, and manufacturing is flat at 5 of 25 (Appendix D). Four sectors have no counterpart among the ten training domains: government, information, wholesale trade and retail trade, 95 tasks in all. They move from 20.0% to 33.7% at the 90% threshold, while the five sectors that overlap the training domains move from 19.2% to 32.0%.

Tasks without PDF inputs. Only 38 tasks have a PDF among their inputs; 87 have other files such as workbooks, Word documents, images and audio, and 95 have none. All three groups gain at the 90% threshold: tasks with PDF inputs from 16% to 42%, tasks with non-PDF inputs from 15% to 21%, and tasks with no input files from 25% to 40%.

4 Changes in tool use and reasoning on GDPval

We compared the two models' use of tools, the text they retrieved from the environment, and the structure of their reasoning across the 94 task pairs in our replication run for which both models produced a gradable deliverable (Table 3; metric definitions in Appendix E).

Table 3: Session behaviour on GDPval, base versus trained, over 94 paired tasks with a gradable deliverable on both sides (60 pairs for source coverage, which needs declared input files). Means unless stated; definitions and paired bootstrap intervals are in Appendix E.

Behaviour	Base model	Trained model
Named source files accessed before deliverable construction	76.1%	**89.9%**
Read-only tool calls per write call	2.77	**1.72**
Environment text retrieved per task (tokens)	13,459	**7,172**
Share of retrieved text that was unique	67.0%	**73.9%**
Reasoning messages per task	26.4	**18.8**
Words per reasoning message	214	**486**
Tasks with an executable post-write reconciliation check	5.3%	**14.9%**

The higher source coverage did not come from reading more. The trained model accessed more of the named source files before it started building while making fewer read-only tool calls, retrieving less text, and repeating less of what it retrieved. The number of calls that write or transform files did not change (7.9 against 8.1 per task), so the reduction is in inspection, not construction. It also performed more multi-unit inventory operations before building, such as enumerating every sheet in a workbook or every page of a PDF, at a rate of about a quarter of a call more per task.

Reasoning was consolidated into fewer, longer messages. The trained model produced fewer reasoning messages per task, but each was more than twice as long, so a single reasoning block more often connected source discovery to the calculations and artifact design that followed. The trained model also more often ran a concrete check on what it had written: the share of tasks with at least one executable post-write consistency or reconciliation check, an expected-versus-actual comparison, a formula audit or a totals check aimed at the deliverable, nearly tripled.

Taken together, the trained model inspected a more complete set of sources with fewer and less repetitive tool calls, and checked its output more often.

5 Paired trajectory examples

We read base and trained trajectories side by side on both benchmarks. The differences fall under four high-level skills that both settings require and that the trained model exercised more often:

- **Understanding what the source material represents.** Reading a table, sheet or list as the records it actually contains, rather than as a sample or a template.
- **Checking whether the evidence in hand covers the whole task.** Establishing the full scope of the job before computing.
- **Recognizing when a missing detail or an inconsistency changes the answer.** Resolving it rather than noting it and continuing.
- **Carrying every requirement of the task through to the final output.** Delivering all requested elements in the requested form.

Each skill appears both when reasoning over a supplied PDF and when using tools to produce a deliverable. Section 5.1 works through the second skill with one task from each benchmark. Table 4 gives one paired example for each of the other three: in each row the GDP.pdf case is a single-turn answer over a supplied document and the GDPval case is a deliverable built with tools, and the base model's failure and the trained model's correction are of the same kind in both.

Table 4: Paired examples of three of the skills listed at the start of Section 5. Each row is one form of a skill, shown in a GDP.pdf answer and in a GDPval deliverable.

Skill and form	GDP.pdf	GDPval
What the source represents *Keeping records complete and distinct*	A five-year executive-pay table contained six records because two executives served in one year. The base model kept only one executive per year in its percentage calculations; the trained model included all six records.	Asked to expand a workstation checklist into an action tracker, the base model supplied one sample issue and blank rows. The trained model carried all 17 checklist items into the tracker, each with action and status fields.
Inconsistencies that change the answer *Keeping schedules consistent*	The base model calculated that a factory production run needed 29 hours but assigned it about five hours in the schedule, dropping a day when converting the end time. The trained model's schedule allowed enough time for the quantities it planned to produce.	An apartment-preparation plan required an extra day for a refrigerator installation. The base model acknowledged this but left the ready date unchanged; the trained model moved it back one day.
Every requirement into the output *Meeting all requested requirements*	A task required filtering a long table by product category and a numeric cutoff. The base model missed qualifying records and included others below the cutoff. The trained model returned all 600 matching records with correct values.	The base model squeezed an investor summary onto one page but left out requested details and cut sentences short. The trained model included the required content in a readable one-page layout.

5.1 Checking whether the evidence covers the whole task

One task from each benchmark illustrates the same skill.

A GDP.pdf task asked for the parts needed to build two complete sets of seaming rolls for a can-seaming machine, from a 40-page service manual. The manual describes six seaming stations, each with two rolls. The base model treated a single station's pair of rolls as a whole-machine set and reasoned, "*Multiply per-roll quantities by 4.*" The trained model established the machine layout first: "*The 61H has six upper seaming chucks ... Two complete machine sets:* $12 \times 2 = 24$ *seaming-roll assemblies*," and scaled the parts list accordingly.

The same kind of scope error appeared in a GDPval workload task. Its sources were Excel workbooks: a stakeholder registry, a March budget, and a timekeeping export with four weekly sheets. No PDFs were involved. The base model opened the first sheet and produced totals for the whole month. It noticed that the result looked wrong:

> "*Everyone is under 40% utilization. That's quite low. Let me verify by checking if maybe the timekeeping data is weekly or something.*"

But it did not perform that check. Instead it decided that the source probably covered only part of the month and continued with the incomplete data. The trained model identified the full month's data,

> "*The timekeeping export has 4 sheets: Mar. 3, Mar. 10, Mar. 17, Mar. 24 ...*"

and combined all four weeks before calculating employee and department workloads and comparing project hours against the budget.

In both tasks the base model treated one part as the whole and started calculating. The trained model first established what the full task covered: all six machine stations, or all four weeks of timekeeping data. The workload task also shows the third skill in its base form: the base model noticed an inconsistency that should have changed its answer and continued without resolving it.

6 Limitations and remaining failures

While the trained model failed some tasks that the base model passed, we did not find one clear regression pattern. Failures had a variety of causes. Several GDPval runs ended without a deliverable or saved one outside the folder the harness collects. One otherwise correct analysis put required mappings in notes rather than in the requested tables. A literature review used constructed references, and a pharmacy guide included made-up drug identifiers.

Failures related to tools, file formats, retry behaviour and delivery mechanics remain. The GDP.pdf training data does not exercise these aspects of the GDPval setting, and most residual GDPval failures on tasks where both models produced a deliverable are of this kind.

Finally, three methodological limits. Each GDPval task was run once per checkpoint, so individual task movements mix checkpoint behaviour with sampling variance; the aggregate and paired statistics are the reliable quantities. Scores come from LLM graders against expert rubrics, and our reading of individual trajectories found grader errors in both directions. And the behavioural metrics of Section 4 are syntax-based proxies computed from tool calls and reasoning text; they show what the model did, not whether it did it for the right reason.

7 Conclusion

Post-training Kimi-K2.7 on 1,465 single-document GDP.pdf tasks raised its full-task pass rate on the held-out GDP.pdf split from 11.0% to 24.5% and raised the share of GDPval tasks scoring at least 90% of rubric points from 19.5% to 32.7%. The GDPval gains were present in sectors outside the training domains and in tasks with no PDF inputs. Trajectory analysis shows that the trained model accessed more of the named sources before building, made fewer and less repetitive read-only tool calls, consolidated its reasoning into fewer and longer messages, and checked its written output more often. Paired examples show the same four skills on both benchmarks: understanding what the source material represents, checking that it covers the whole task, recognizing when a missing detail or inconsistency changes the answer, and carrying every requirement through to the final output.

Kimi-K2.7 could already use the tools that GDPval requires; training changed how it organized that tool use around the task. The results indicate that post-training on demanding single-document tasks can change behaviour in a tool-using setting that differs from training in interaction mode, input format and domain coverage, without reproducing that setting in the training data. Remaining failures on dense visual pages, on pages beyond the image budget, and in workflow mechanics identify where further training data would be needed.

References

Suhaas Garre, Emily Ritchie, Sushant Mehta, and Edwin Chen. GDP.pdf: Benchmarking grounded multimodal reasoning over professional PDF documents. *arXiv preprint arXiv:2607.11192*, 2026. URL https://arxiv.org/abs/2607.11192. Presented (non-archival) at the 2nd Workshop on Knowledge-Intensive Multimodal Reasoning (KnowledgeMR) at CVPR 2026.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2507.17746.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2106.09685.

Kimi Team. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025. URL https://arxiv.org/abs/2507.20534.

Kevin Lu and Thinking Machines Lab. On-policy distillation. Thinking Machines Lab: Connectionism (blog). https://thinkingmachines.ai/blog/on-policy-distillation/, October 2025. Accessed September 2026.

Miles Team. Miles: Enterprise-grade reinforcement learning for large-scale model post-training. GitHub repository. https://github.com/radixark/miles, 2026. Accessed September 2026.

Moonshot AI. Kimi K2.7 Code. Hugging Face model card. https://huggingface.co/moonshotai/Kimi-K2.7-Code, 2026. Coding-focused agentic model built on Kimi K2.6; open weights released June 2026. Accessed September 2026.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, Natalie S. Kim, Patrick Chao, Samuel Miserendino, Gildas Chabot, David Li, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. GDPval: Evaluating AI model performance on real-world economically valuable tasks. *arXiv preprint arXiv:2510.04374*, 2025. URL https://arxiv.org/abs/2510.04374.

John Schulman and Thinking Machines Lab. LoRA without regret. Thinking Machines Lab: Connectionism (blog). https://thinkingmachines.ai/blog/lora/, September 2025. Accessed September 2026.

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. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019. URL https://arxiv.org/abs/1909.08053.

UK AI Security Institute. Inspect AI: Framework for large language model evaluations. Software. https://inspect.aisi.org.uk/, 2024. Released May 2024. Source: https://github.com/UKGovernmentBEIS/inspect_ai. Accessed September 2026.

UK AI Security Institute, Arcadia Impact, and Vector Institute. Inspect Evals. Software. https://ukgovernmentbeis.github.io/inspect_evals/, 2024. A repository of community-contributed LLM evaluations for Inspect AI, announced November 2024. Source: https://github.com/UKGovernmentBEIS/inspect_evals. Accessed September 2026.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. Checklists are better than reward models for aligning language models. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025. URL https://arxiv.org/abs/2507.18624.

Feng Yao, Liyuan Liu, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. Your efficient RL framework secretly brings you off-policy RL training. Blog post. https://fengyao.notion.site/off-policy-rl, August 2025. Accessed September 2026.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026. URL https://arxiv.org/abs/2601.18734.

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. *arXiv preprint arXiv:2507.18071*, 2025. URL https://arxiv.org/abs/2507.18071.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024. URL https://arxiv.org/abs/2312.07104.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An LLM post-training framework for RL scaling. GitHub repository. https://github.com/THUDM/slime, 2025. Accessed September 2026.

A Training details

Model and adapters. Base model Kimi-K2.7 [Moonshot AI, 2026, Kimi Team, 2025]; LoRA [Hu et al., 2022] rank 32, $\alpha = 32$, dropout 0, applied to the attention projections (`q_a_proj`, `q_b_proj`, `kv_a_proj_with_mqa`, `kv_b_proj`, `o_proj`) and MLP projections (`gate_proj`, `up_proj`, `down_proj`). LoRA rather than full fine-tuning follows Schulman and Thinking Machines Lab [2025]. Training used 64 H200 GPUs with Megatron-based parallelism [Shoeybi et al., 2019], SGLang for rollouts [Zheng et al., 2024], and the slime [Zhu et al., 2025] and Miles [Miles Team, 2026] frameworks.

Stage 1: on-policy self-distillation. 600 tasks, balanced across domains. For each task a strong model (GPT-5.6 Sol) wrote targeted feedback derived from the rubric, stating which criteria a prior trace missed and why, without stating the answer; the feedback is private. The student generates from the public prompt only. The teacher is the same base model with the same LoRA adapter, prompted with the public prompt plus the private feedback, and scores the exact student token sequence by a prefill-only request. The per-token signal is $-\beta\,(\log \pi_{\text{student}} - \log \pi_{\text{teacher}})$, a reverse-KL estimator following Lu and Thinking Machines Lab [2025], Zhao et al. [2026]; the scalar task reward is zero. Because the teacher prompt is longer, the student's response budget is reduced by the length of the private prompt before generation, and samples whose private prompt plus response would exceed the window are rejected rather than truncated.

Stage 2: reinforcement learning. 865 tasks, disjoint from stage 1, balanced across domains. GSPO [Zheng et al., 2025] with sequence-level importance ratios (clip 0.20/0.28) and group-relative advantages [Shao et al., 2024]; 64 prompts $\times$ 8 samples per update; learning rate 10^{-5}; Adam $(0.9, 0.98)$; weight decay 0.1; gradient clipping 0.1; no KL or entropy terms; truncated importance sampling [Yao et al., 2025]; temperature 0.7; top-p 0.95; context 65,536 tokens; truncated samples dropped; maximum reasoning effort. Reward: the fraction of rubric criteria judged satisfied by GPT-5.4 Mini (medium reasoning) using the benchmark's rubric verifiers. The training configuration specifies page dropping and a character budget on the extracted text to fit documents into the window.

B Input construction and evaluation harness

GDP.pdf. Each task is rendered as a system prompt ("expert document analyst": search all supplied pages before concluding that information is absent; use the relevance guide only for navigation; for visual questions, locate the labelled feature and trace its boundary), followed by a user turn containing the task, a prompt-derived relevance guide of likely-relevant page numbers with short excerpts, the full extracted text of every page (native text where present, OCR otherwise), and rendered page images selected by the same relevance heuristic: up to sixteen pages at roughly 4,096 tokens each. Documents of at most sixteen pages are fully imaged; of the 64 public documents longer than that, 54 received sixteen images and 10 received between four and fifteen. At evaluation the model is served in a 262,144-token window with a 65,536-token response reservation, so inputs up to 196,608 tokens keep the full response budget; for the two longer inputs the reservation is halved once or twice. On the public split, input tokens run from 6,359 to 239,591 (median 87,493; mean 91,283), with a median of 23K text tokens and a mean image share of 66%; 72 tasks exceed 65,536 tokens and 19 exceed 131,072. Trajectories without a completion or grade (3 of 100 base, 6 of 400 trained) count as failures.

The public leaderboard supplies the native PDF to each model's API. Under that protocol the base model scores 16% (which would rank 24th), against 11.0% under our text-plus-images construction. Our leaderboard positions are computed by inserting our scores into the September 12, 2026 snapshot of the public leaderboard, counting ties as the higher rank.

GDPval. The 220 gold tasks in the Inspect Evals implementation [UK AI Security Institute et al., 2024], with Python and shell tools in a sandbox. Input files are placed in a `reference_files` folder that the model must discover; deliverables are written to `deliverable_files`; the final assistant message is graded together

with the files by GPT-5.4 Mini (medium reasoning) against the task rubric. Prompts are short (median 2.4K characters); the long context of a session is generated by the model's own tool use. Base and trained models were evaluated with identical harness settings and per-session limits within each comparison run.

C GDP.pdf results by input length

Table 5: GDP.pdf public split by input-length bucket. Trained columns are means over four runs (range in parentheses for pass rate). The dashed boundary at 65,536 tokens is the training context.

		Full-task pass (%)		Criteria pass rate	
Input tokens	n	Base	Trained	Base	Trained
<32K	12	0.0	12.5 (0–25)	0.588	0.643
32–64K	16	31.2	46.9 (44–50)	0.771	0.816
64–128K	53	7.5	18.4 (17–21)	0.653	0.715
>128K	19	10.5	30.3 (26–32)	0.631	0.710
≤64K	28	17.9	32.1	0.693	0.742
>64K	72	8.3	21.5	0.647	0.714

The difference in gain between inputs above and below the training context is -1.1 pp in full-task pass rate (permutation $p = 0.91$; 95% bootstrap interval $[-13.0, +9.7]$), so there is no evidence that the gain attenuates with length. The mean criteria pass rate over all 100 tasks rises from 0.660 to 0.722 (Wilcoxon $p = 0.001$).

D GDPval results by sector

Table 6: GDPval by sector (25 tasks each; retail 20). Tasks at or above 90% of rubric points and paired improved/regressed counts, base → trained. Starred sectors have no counterpart among the ten training domains.

Sector	≥ 90% (base)	≥ 90% (trained)	Improved / regressed
Finance and insurance	5	10	16 / 3
Health care and social assistance	7	9	15 / 3
Real estate and rental and leasing	2	8	11 / 8
Government*	4	11	16 / 8
Professional, scientific and technical	5	8	14 / 7
Information*	6	8	13 / 9
Retail trade*	6	8	9 / 8
Manufacturing	5	5	12 / 9
Wholesale trade*	3	5	13 / 8
All	43	72	119 / 63

E Behavioural metric definitions

All metrics in Table 3 are computed from the saved trajectories of the 100-task replication run, on the 94 task pairs for which both models produced a gradable deliverable. Paired changes carry 95% percentile-bootstrap intervals over tasks.

Named source files accessed before deliverable construction.

The fraction of the task's declared input files whose name appears in a source-reading tool call before or

in the same call as the first write to the deliverables folder; a loop that reads every file in the input folder counts as covering all of them. Computed on the 60 pairs whose task declares input files. Paired change +13.8 pp, interval [+4.9, +23.6].

Read-only tool calls per write call.
Tool calls are classified from the tool name and the executed shell or Python. A call is *write or mixed* if it contains an observable mutation (creating, saving, converting, copying, deleting or installing); all other calls are *read-only*. The ratio is computed per trajectory and averaged. Read-only calls fall by 6.5 per task (interval [−9.9, −3.4]); write-or-mixed calls change by +0.2 with an interval spanning zero.

Environment text retrieved.
Tokens (`cl100k_base`) in textual tool outputs, after removing image, audio, binary and base64-like payloads. Paired change −6,287 tokens (interval [−10,230, −2,817]); the median change is about −21 tokens.

Share of retrieved text that was unique.
Unique tokens divided by retrieved tokens, where a token is unique if it is not covered by a repeat of an exact 12-token span seen earlier in the same output or in an earlier output of the same trajectory. Paired change +6.9 pp (interval [+3.1, +10.6]).

Reasoning messages and words per message.
The count of assistant messages containing reasoning text, and total reasoning words divided by that count. Reasoning messages fall by 7.6 per task while reasoning words rise by about 3,460.

Executable post-write reconciliation check.
Whether any read-only tool call that targets the deliverables folder after the first write runs an expected-versus-actual comparison, a formula audit, a balance check, or a total or count consistency check. Presence per task; 5 of 94 base tasks against 14 of 94 trained tasks (interval on the paired change [+2.1, +17.0] pp).