On-Policy Self-Distillation via Prompt Optimization

Jongho Park, Donghyun Lee, Matei Zaharia, Jason D. Lee
UC Berkeley
{jjhpark, lukedhlee, matei, jasondlee}@berkeley.edu

Abstract

Reinforcement Learning (RL) has become the dominant paradigm for post-training language models (LMs), yet its scalar reward collapses rich environment feedback into a single number. On-policy self-distillation (OPSD) recovers some of this signal by training a student to match a stronger self-teacher that is given privileged information. On the other hand, parameter-free prompt optimization methods like GEPA excel in domains that offer rich feedback and use compound LM programs.

In this paper, we bridge this divide by introducing **Self-Distillation via Prompt Optimization** (SDPRO), a framework that jointly optimizes both prompts and parameters. SDPRO leverages environment feedback through an iterative two-phase cycle: it first discovers generalizable prompts via GEPA, and then OPSD internalizes those gains into the model weights. Not only does SDPRO lead to better performance than GEPA alone, but our method naturally extends OPSD to compound LM programs, which prior OPSD formulations could not naturally handle. On HotpotQA with Qwen3-8B, three cycles outperform GEPA by +4.7pts and GRPO by +22.5pts under the same compute budget, with no degradation and even gains on the held-out HoVer and IFBench benchmarks, which use distinct compound programs. We present these results as a proof of concept that integrating prompt and parameter optimization is a promising route to learning from rich natural-language environment feedback.

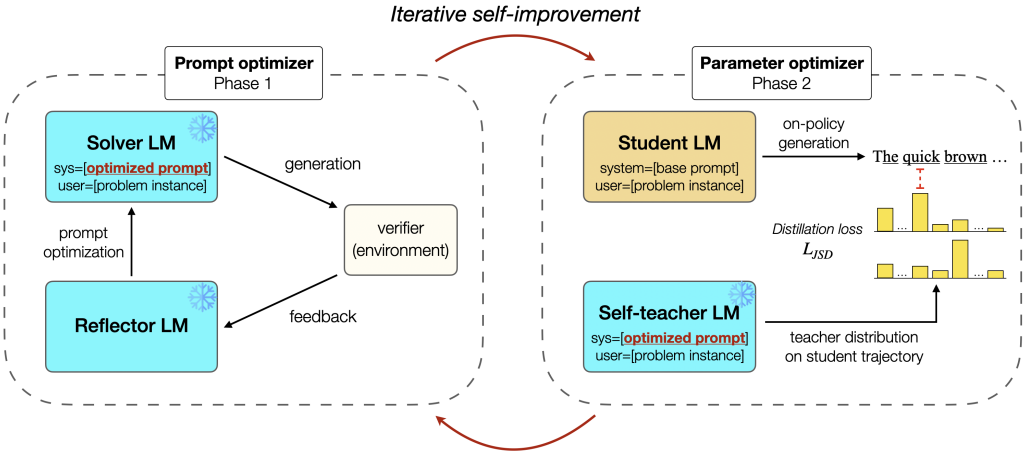

Figure 1: **Overview of SDPRO.** One iteration alternates between a Phase 1 prompt optimizer (GEPA reflectively evolves the instructions of a multi-module DSPy program against environment feedback) and a Phase 2 parameter optimizer (a frozen *self-teacher* conditioned on the GEPA-optimized prompts supplies a token-level distillation signal to a *student* conditioned on the base prompts, on student-sampled trajectories, via a JSD loss). The student's LoRA update is merged into the base model; the next iteration starts GEPA from the new weights, so prompts and parameters co-evolve.

43rd International Conference on Machine Learning (ICML 2026). RLxF: Reinforcement Learning from World Feedback.

1 Introduction

Reinforcement learning (RL) has become the default recipe for post-training language models (LMs) [Ouyang et al., 2022, Shao et al., 2024], yet its scalar trajectory reward discards any dense textual feedback from its environment. In response, on-policy self-distillation (OPSD) has emerged as a promising alternative [Zhao et al., 2026, Hübotter et al., 2026]: train a student to match a stronger *self-teacher* at the token level on the student's own rollouts, where the teacher's advantage comes from privileged information injected at training time but withheld at deployment. Because this context is expressed in natural language, OPSD absorbs exactly the kind of dense feedback RL cannot.

The effectiveness of OPSD therefore hinges on what the self-teacher is conditioned on, and prior work has explored a range of choices: gold solutions [Zhao et al., 2026], successful rollouts, environment feedback [Hübotter et al., 2026], and the model's own reflections on failure [Zhang et al., 2026]. These choices share two properties that limit their generality. First, each is tied to a single training example: a gold answer or a successful rollout is example-specific and provides little signal that generalizes across the dataset. Second, they are designed for the single-call, single-prompt setting, and do not extend cleanly to compound LM programs: multi-module pipelines that chain several LLM calls with distinct roles.

Compound LM programs—agentic pipelines, multi-step reasoners, and DSPy [Khattab et al., 2023] programs—have become the dominant form of modern LM systems precisely because their modular structure absorbs the dense textual feedback that scalar rewards discard: prompt optimizers like GEPA [Agrawal et al., 2025] reflectively rewrite each module's instructions against environment signal, assigning credit per module. Yet this is exactly where existing OPSD teacher contexts fall apart. A gold trajectory specifies one coherent path, but a student's on-policy rollout diverges as soon as its first module output differs from the gold's. Every subsequent module then conditions on intermediate states the gold never encountered, and the teacher describes a downstream problem the student is no longer solving. Thus, token-level supervision conditioned on the gold degrades to first-module supervision, leaving the rest of the program without per-module signal.

In turn, we introduce **Self-Distillation via PRompt Optimization (SDPRO)**, which constructs the teacher context through prompt optimization that generalizes across examples and jointly optimizes parameters and per-module prompts in compound LM programs, directly addressing the dataset-level and per-module limitations above. In doing so, it turns OPSD into a self-improvement loop where each SDPRO cycle has two steps. First, we run GEPA that returns strong prompts that apply across multiple examples. Second, we use the resulting prompts as the privileged context for the self-teacher and distill the base prompt student on its own on-policy traces, folding the prompt-discovered behavior into the weights. The cycle then repeats from a now-stronger student, so prompts and weights coevolve and gains compound across iterations rather than saturating after GEPA.

On HotpotQA with Qwen3-8B, three SDPRO iterations exceed GEPA alone by $+4.7$ points and GRPO by $+22.5$ points under a matched rollout budget. We further show that the prompts SDPRO discovers are tuned to the updated weights, not the base model—evidence that prompts and parameters co-adapt rather than improve independently. Finally, the best HotpotQA checkpoint transfers cleanly to the held-out HoVer, IFBench, and PUPA benchmarks, each of which uses a distinct compound pipeline, with no degradation and in fact gains of $+9.0$ points on HoVer and $+3.2$ points on IFBench. Together, these results suggest that prompt and parameter optimization is a promising route to learn from rich natural-language environment feedback for compound LM programs.

2 SDPRO: Co-evolving Prompts and Parameters

In this section, we describe our algorithm for SDPRO, which proceeds in an iterative two-phase cycle of prompt optimization and subsequent parameter optimization. We defer the discussion of prior related work to Appendix A.

Let π_{θ_i} be the LLM policy with weights θ_i at iteration i, and let each module of a multi-module program carry a natural language base prompt P_0. Denote the best prompt found at iteration i for example x as $\mathcal{P}_i^\star(x)$ and the aggregate-best prompt for all calibration examples as P_i^*. We divide the training set into a GEPA-train set and GEPA-calibration set. Figure 1 summarizes SDPRO.

Phase 1 (Prompt optimization). We run GEPA on π_{θ_i} to obtain a set of best prompts. Throughout GEPA's inner loop, the Reflector LM analyzes rollouts from sampled prompts and proposes improved

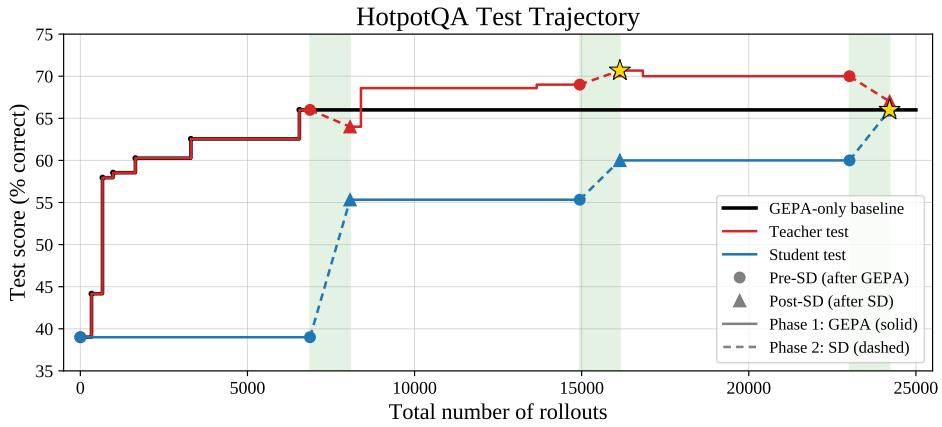

Figure 2: **HotpotQA test accuracy.** SDPRO escapes GEPA's plateau on two fronts: GEPA finds progressively stronger prompts once the weights have been distilled ($66 \rightarrow 69 \rightarrow 70$), and self-distillation absorbs those prompt-discovered behaviors into the weights so that the student that never sees a GEPA prompt at inference catches up to the GEPA-only teacher under base prompts alone ($39 \rightarrow 66$). Together, the two phases drive both the teacher ($+4.7$ over GEPA-only) and the student ($+23.5$ over base) past what either prompt or parameter optimization achieves in isolation.

candidates that distill insights from the environment's feedback. Crucially, we do *not* collapse GEPA's search to a single aggregate-best prompt. Instead, we keep its entire *Pareto front*–the set of non-dominated candidate prompts, each of which is the strongest prompt for some subset of calibration examples. For every calibration example x, we record the Pareto-front candidate $\mathcal{P}_i^\star(x)$ that scores highest on x, yielding a per-example teacher map rather than one global teacher. We use the same model parameters for both prompt rollouts and prompt improvement. See Agrawal et al. [2025] for further implementation details for GEPA.

Phase 2 (Parameter optimization). We distill this per-example Pareto teacher into the weights using on-policy self-distillation. For each example x, the teacher $(\pi_{\theta_i}, \mathcal{P}_i^\star(x))$—the frozen model θ_i conditioned on *that example's* best prompt—supplies the target distribution, while the student $(\pi_{\theta_0+\Delta_i}, \mathcal{P}_{\text{base}})$ uses the short base prompts with a LoRA adapter Δ_i, which is initialized as Δ_{i-1}. The student runs the program end-to-end to produce on-policy traces, and at each module invocation, we minimize the Jensen-Shannon divergence (JSD) between the student's and the teacher's next-token distributions over the student completion [Agarwal et al., 2024].

Because every example is paired with its own best teacher rather than one global prompt for each module of the compound LLM, the distillation signal preserves the full diversity of GEPA's discovered behaviors instead of regressing to the single highest-scoring compromise prompt. Finally, $\theta_i \leftarrow \theta_0 + \Delta_i$ as we only keep one LoRA adapter through multiple iterations of SDPRO. We then cycle back to Phase 1 to re-run prompt optimization from the stronger weights, so prompts and parameters co-evolve across training. For evaluation as seen in Figure 2, after Phase 1 iteration i, we evaluate the new aggregate-best prompt P_i^* on weight θ_i. After Phase 2 iteration i, we evaluate both the new student θ_{i+1} and the teacher with the previous aggregate-best prompt P_i^*.

3 Experiments

Evaluation. We train and evaluate SDPRO on HotpotQA [Yang et al., 2018] using Qwen3-8B as both student and self-teacher. Following GEPA's protocol, the inference program is a *multi-module* DSPy [Khattab et al., 2023] pipeline of four `ChainOfThought` modules (`summarize1`, `create_query_hop2`, `summarize2`, `final_answer`) backed by a fixed ColBERTv2 retriever; GEPA optimizes the instructions of every module. This is in contrast to prior OPSD work [Zhao et al., 2026, Hübotter et al., 2026], which optimizes a single prompt for a single inference call, a regime that misses the cross-module coordination that gives modern LM programs their accuracy. Splits are $150/300/300$ (train/val/test) and we report held-out test exact-match accuracy.

Table 1: **HotpotQA with Qwen3-8B.** Accuracy on the test set from 25,000 rollouts. Teacher uses the GEPA-optimized prompt, and Student uses a base prompt. Base GEPA consumes the total rollout budget, while each GEPA phase in SDPRO uses 6,871 rollouts. SDPRO escapes the GEPA's ceiling through iterative prompt-weight updates.

Method	Phase	Teacher P_i^*	Student P_0
GEPA	—	66.0	42.3
GRPO	—	—	43.5
SDPRO	iter-0 post-GEPA (θ_0)	66.0	—
	iter-0 post-SD (θ_1)	64.0	55.3
	iter-1 post-GEPA (θ_1)	69.0	—
	iter-1 post-SD (θ_2)	**70.7**	60.0
	iter-2 post-GEPA (θ_2)	70.0	—
	iter-2 post-SD (θ_3)	67.0	**66.0**

Table 2: **Performances of best GEPA prompts in isolation.** Applying the best prompts found in each iteration on the base model yields worse scores. This shows that SDPRO does not simply find better prompts, but that optimizing prompts and parameters works in tandem.

Prompt	**Qwen3-8B**	**Model**
iter-1 P_1^*	θ_0 : 63.7	θ_1 : 69.0
iter-2 P_2^*	θ_0 : 61.0	θ_2 : 70.0

SDPRO setup. We run three total SDPRO GEPA→Self-Distillation (SD) loops. Each Phase 1 calls GEPA with a $6{,}871$-rollout budget, while each Phase 2 trains the same LoRA adapter with rank r=64 and α=128 for 25 steps with the JSD loss. We approximate the distribution using the top-k tokens. We compare against two approaches under a matched rollout budget: **(1)** GEPA for the frozen Qwen3-8B with 25,000 rollouts, **(2)** multi-module GRPO training as done in Agrawal et al. [2025].

Results. Table 1 reports HotpotQA test accuracy. SDPRO improves over GEPA alone at every iteration and over the base model by a wide margin. After three iterations, the teacher reaches $\mathbf{70.7}$ ($+4.7$ over GEPA alone, $+28.4$ over base), and the student–which only sees the base prompts–reaches 66.0 ($+23.7$ over base) by internalizing the best prompts. We highlight four findings in particular.

(i) GEPA on stronger weights finds stronger prompts. In Table 1, the post-GEPA score (GEPA's discovered teacher *before* any SD at that iteration) climbs from 66 at iter-0 to 70.0 at iter-2 post-GEPA. The same GEPA budget produces a meaningfully better prompt set once the base model has been distilled: weight updates expand the space of behaviors GEPA can elicit.

(ii) Self-distillation internalizes prompt-discovered behavior. The student never sees a GEPA prompt at inference time, yet SD lifts its score from 42.3 to 66.0. A $+23.7$-point gain that lives entirely in the weights. The multi-module GEPA program's behavior is thus being absorbed into the model's parameters rather than remaining bound to the prompt boundary.

(iii) Prompts and parameters contribute multiplicatively. In Table 2, the iter-2 prompts applied to the base weights score 61.0, while the iter-2 weights with the iter-2 prompts score 70.0; the SD-only contribution on top of GEPA prompts is thus $+9$ points, and it stacks with the $+27.7$-point prompt contribution ($42.3 \rightarrow 70.0$). Neither component on its own reaches the joint optimum.

(iv) SDPRO student's learning transfers to out-of-distribution tasks. Table 4 shows that the best HotpotQA student improves over the Qwen3-8B base by +9.0 on HoVer [Jiang et al., 2020] and +3.2 on IFBench [Srivastava et al., 2024], with only a 2.6 point drop on PUPA [Siththaranjan et al., 2024].

4 Conclusion

We introduced SDPRO, which recasts prompt optimization as a mechanism for constructing the privileged context an on-policy self-teacher needs—extending OPSD to compound LM programs that prior teacher-context choices could not reach. The two phases are complementary: GEPA discovers behaviors that the environment that apply across multiple LM modules, and self-distillation internalizes those behaviors so the next GEPA round explores from a stronger model. We view this as preliminary evidence that prompt and parameter optimization should be co-designed rather than studied as orthogonal axes in multi-module pipelines. We discuss the limits of the current method–in particular, benchmarks where SDPRO does not exceed GEPA alone–in Appendix B.

Acknowledgments and Disclosure of Funding

We would like to thank Jongwon Jeong and Alex Dimakis for helpful discussions and feedback. Furthermore, this work is supported in part by VESSL AI for providing valuable support in GPU infrastructure and resources.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pages 21246–21263, 2024.

Lavanya Aryan Agrawal et al. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. HoVer: A dataset for many-hop fact extraction and claim verification. *arXiv preprint arXiv:2011.03088*, 2020.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Anand Siththaranjan et al. PAPILLON: Privacy preserving inference via local models and untrusted apis. *arXiv preprint arXiv:2410.17127*, 2024.

Dilara Soylu, Christopher Potts, and Omar Khattab. Fine-tuning and prompt optimization: Two great steps that work better together. *arXiv preprint arXiv:2407.10930*, 2024.

Saurabh Srivastava et al. IFBench: Instruction-following benchmark. *arXiv preprint arXiv:2406.00000*, 2024.

Rishabh Tiwari, Kusha Sareen, Lakshya A. Agrawal, Joseph E. Gonzalez, Matei Zaharia, Kurt Keutzer, Inderjit S. Dhillon, Rishabh Agarwal, and Devvrit Khatri. Learning, fast and slow: Towards LLMs that adapt continually. *arXiv preprint arXiv:2605.12484*, 2026.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Yuwei Zhang, Sha Li, Changlong Yu, Qin Lu, Shuowei Jin, Chengyu Dong, Haoran Liu, Ilgee Hong, Xintong Li, Zhenyu Shi, Bing Yin, and Jingbo Shang. Learning with rare success but rich feedback via reflection-enhanced self-distillation, 2026. URL `https://arxiv.org/abs/2605.12741`.

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.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations (ICLR)*, 2023.

Noah Ziems, Dilara Soylu, Lakshya A Agrawal, Isaac Miller, Liheng Lai, Chen Qian, Kaiqiang Song, Meng Jiang, Dan Klein, Matei Zaharia, et al. Multi-module grpo: Composing policy gradients and prompt optimization for language model programs. *arXiv preprint arXiv:2508.04660*, 2025.

A Related Work

Prompt optimization. A growing literature optimizes natural-language prompts directly. APE [Zhou et al., 2023] and OPRO [Yang et al., 2024] search single-prompt instructions with an LLM in the loop. MIPRO [Opsahl-Ong et al., 2024] jointly searches instructions and few-shot demonstrations for DSPy [Khattab et al., 2023] programs. GEPA [Agrawal et al., 2025] extends this to reflective, per-module instruction search over compound LM programs and is the optimizer SDPRO instantiates. Across this line of work, the discovered prompts are the final artifact deployed at inference. SDPRO instead treats them as a transient training signal that is folded into student weights, so deployment uses the short base prompt.

On-policy self-distillation. Classical knowledge distillation [Hinton et al., 2015] on teacher-generated text suffers from distribution mismatch when the student must later generate its own sequences. MiniLLM [Gu et al., 2024] and Generalized Knowledge Distillation (GKD) [Agarwal et al., 2024] distill on *student-generated* traces with reverse-KL or generalized-JSD objectives, establishing on-policy distillation as the dominant paradigm for LM compression. A recent line of work then removes the teacher–student weight asymmetry: the teacher shares the student's weights and gains its advantage solely from privileged context unavailable at deployment. OPSD [Zhao et al., 2026] conditions the self-teacher on the gold answer with a clipped JSD objective; SDPO [Hübotter et al., 2026] replaces it with environment feedback and execution traces. SDPRO inherits this on-policy, same-weights setup but argues the privileged context should itself be the output of an optimizer rather than a fixed source.

Joint prompt and parameter optimization. A growing set of methods optimize prompts and weights together rather than treating them as separate concerns. BetterTogether [Soylu et al., 2024] alternates prompt search and supervised fine-tuning for DSPy programs, showing the two stages are complementary. Multi-module GRPO [Ziems et al., 2025] composes per-module policy gradients with prompt optimization for compound LM programs, scaling joint optimization to multi-call pipelines. Concurrent work, Fast-Slow Training (FST) [Tiwari et al., 2026], treats optimized context as fast weights alongside GRPO-updated slow weights. Though similar in that both methods utilize prompt optimization, SDPRO differs in two ways. First, we integrate prompt optimization with on-policy self-distillation rather than policy-gradient RL, inheriting the dense token-level supervision of distillation. Second, FST retains the optimized prompts at deployment as a persistent task-specific channel, which presupposes a mechanism for selecting the right prompt at test time, whereas SDPRO distills prompt-discovered behavior directly into the student weights, eliminating need for any routing. Our held-out transfer results in Appendix B illustrate this advantage: the SDPRO student generalizes to new compound pipelines without carrying along the prompts that produced it.

Self-improvement and bootstrapping. Iterated self-improvement loops generate data with the current model, filter or annotate it, and retrain. STaR [Zelikman et al., 2022] bootstraps reasoning by re-training on rationales the model itself produced for correct answers; ReST [Gulcehre et al., 2023] formalizes the generate-filter-retrain cycle; self-rewarding models [Yuan et al., 2024] use the model as its own preference judge; Reflexion [Shinn et al., 2023] uses natural-language failure analyses to drive subsequent attempts. SDPRO is in this family but alternates between two distinct optimization spaces: prompt space via GEPA and weight space via on-policy distillation.

B Results on Additional Tasks

Beyond HotpotQA, we ran SDPRO on four additional Qwen3-8B benchmarks under the same protocol: IFBench [Srivastava et al., 2024] (instruction following), HoVer [Jiang et al., 2020] (multi-hop fact verification), and PAPILLON (PUPA) [Siththaranjan et al., 2024] (privacy-preserving inference). Table 3 reports the best teacher SDPRO test accuracy reached across iterations against the GEPA-only baseline. IFBench and HoVer essentially tie GEPA-only on test, and PUPA gains only marginally, indicating that the SD phase can overfit the GEPA-discovered prompts on these benchmarks.

Limitations. The SD phase relies on the GEPA teacher being meaningfully stronger than the base model along the verifier's metric. When that gap is small (PAPILLON), or when the verifier's

Table 3: **Best SDPRO vs. GEPA-only across additional benchmarks (Qwen3-8B).** Held-out test accuracy (%). Each row is the best teacher score reached over 1–3 SDPRO iterations against the iter-0 PreSD GEPA-only teacher. Numbers in parentheses are the absolute change vs. GEPA-only. SDPRO is a clear win on HotpotQA (from Table 1); it is roughly a tie on IFBench, HoVer, and PUPA.

Benchmark	Base	GEPA only	SDPRO teacher	SDPRO student
HotpotQA	42.3	66.0	**70.7** (+4.7)	66.0 (+0.0)
IFBench	36.9	38.6	37.1 (-1.5)	39.3 (+0.7)
HoVer	35.3	53.3	53.7 (+0.4)	52.0 (-1.3)
PUPA	80.8	91.9	90.4 (-1.5)	90.3 (-1.6)

Table 4: **Out-of-domain evaluation using base prompts only.** The best student checkpoint selected on HotpotQA is evaluated on Hover, IFBench, and PUPA without GEPA teacher prompts.

Evaluation setting	Base Qwen3-8B	Best HotpotQA student	Change
Hover	35.3	44.3	**+9.0**
IFBench	36.9	40.1	**+3.2**
PUPA	80.8	78.2	**-2.6**

reward signal saturates fast (IFBench), there is little for self-distillation to internalize and we observe iteration-to-iteration regressions in the student. We see the strongest SDPRO gains in tasks where the multi-module program produces structured intermediate state (multi-hop reasoning on HotpotQA)– consistent with the claim that SDPRO's benefit grows with the gap between the base model and the GEPA-discoverable program. Still, we find that distilling prompt behavior into the student all the way up to GEPA-only accuracy is possible.

C Experimental details

Table 5 lists the full hyperparameter configuration used for the HotpotQA experiments in Section 3; the additional benchmarks in Appendix B use the same configuration unless noted. The pipeline uses DSPy [Khattab et al., 2023] for the multi-module program, Arbor for trainer/inference orchestration (ZMQ-coupled LoRA training and vLLM serving with per-step weight sync), and HuggingFace TRL's `Trainer` as the underlying training loop. Both teacher and student share the same underlying Qwen3-8B weights and differ only in (a) the per-module instructions placed in the system message and (b) whether the LoRA adapter is active or frozen: the teacher uses the GEPA-optimized (Pareto-front) instructions with LoRA *frozen*, and the student uses the base instructions with LoRA *trainable*.

HotpotQA results. We attach Figure 3 to show both validation and test accuracy trajectories for Qwen3-8B.

Table 5: **Hyperparameters for SDPRO on HotpotQA (Qwen3-8B).**

Group	Hyperparameter	Value
Model & program	Base model	Qwen3-8B
	Program	DSPy `HotpotMultiHop`, 4 `ChainOfThought` modules
	Retriever	ColBERTv2, top-k passages, k=7
	Max context length	8192 tokens
Phase 1 (GEPA)	Reflective search budget	6,871 metric calls / iteration
	Teacher selection	per-example Pareto front
Phase 2 (Self-distillation)	LoRA rank r	64
	LoRA α	128
	LoRA dropout	0.05
	LoRA target modules	$\{q, k, v, o, \text{up}, \text{down}, \text{gate}\}_{\text{proj}}$
	SD steps / iteration	25
	Optimizer	AdamW, bf16
	Learning rate	1×10^{-5}, constant schedule with warmup
	Examples per step / rollouts	4 examples, 12 rollouts each
	Sampling temperature	0.6
	JSD details	symmetric β=0.5, temperature T=2.0, top-k distillation
	Checkpoint selection	best adapter by calibration accuracy, then merged into θ
Iteration & hardware	SDPRO iterations	3
	Weight handling	use one LoRA through iterations
	Hardware	2× H200 NVL (143 GB): one trainer, one vLLM 0.19 server

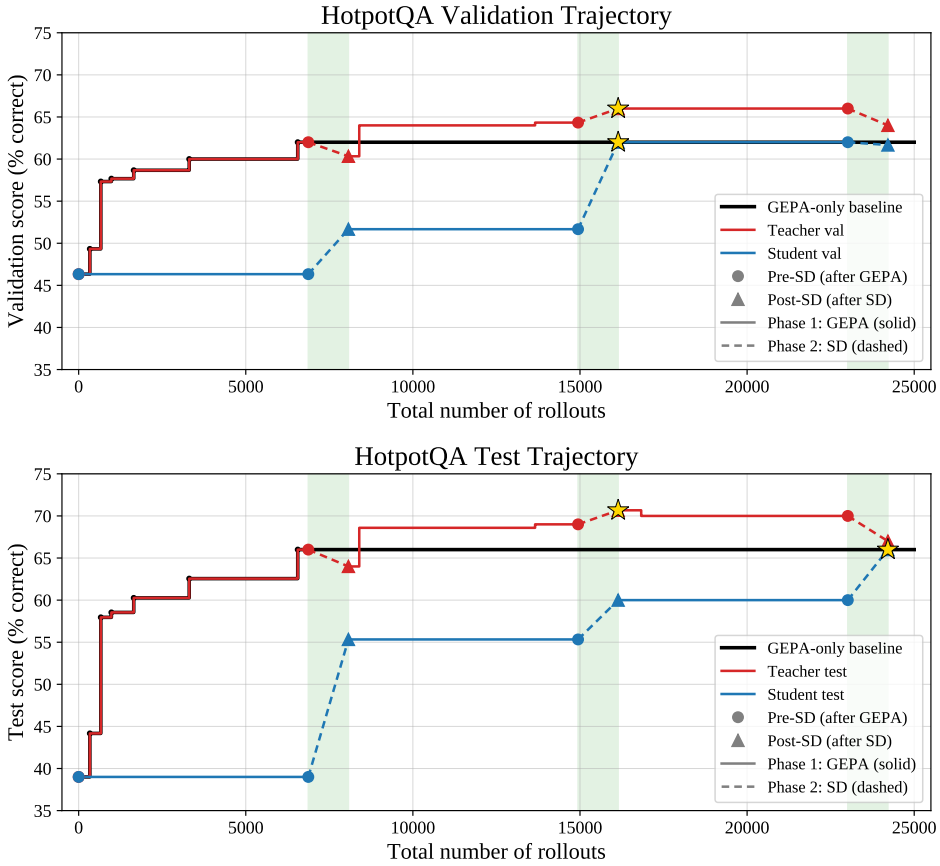

Figure 3: **HotpotQA validation and test accuracy.** SDPRO escapes GEPA's plateau on two fronts: GEPA finds progressively stronger prompts once the weights have been distilled (Pre-SD teacher: $66 \rightarrow 69 \rightarrow 70$), and self-distillation absorbs those prompt-discovered behaviors into the weights so that the student—which never sees a GEPA prompt at inference—catches up to the GEPA-only teacher under base prompts alone ($39 \rightarrow 66$). Together, the two phases drive both the teacher ($+4.7$ over GEPA-only) and the student ($+22.5$ over base) past what either prompt or parameter optimization achieves in isolation. While we see that teacher validation checkpoint and test checkpoint match, this is not the case for the student.