When Does Specialist RL Help a Small-Model Multi-Agent Pipeline? A Cross-Benchmark Study on Long-Document QA

Mert Karabiyik (Solo) mertk@stanford.edu

CS224R Spring 2026 Custom Project

Extended Abstract

Multi-agent pipelines of small language models (SLMs) are an attractive way to do long-document question answering cheaply, but they trail frontier models by a wide margin. This project asks whether reinforcement learning can close that gap, and if so, where in the pipeline it pays off. One would assume the place to spend RL is the synthesizer that actually writes the final answer, but the experiments point somewhere else. I start from MoSA (Karabiyik et al., 2026), a five-agent pipeline of Llama-3.2-3B QLoRA adapters (orchestrator, extractor, calculator, reasoner, synthesizer), and study RL fine-tuning of its specialists with RLOO+, a from-scratch RLOO/GRPO-style trainer I wrote on top of HuggingFace generation. RLOO+ uses a leave-one-out advantage (Ahmadian et al., 2024), per-task reward standardization (Shao et al., 2024), dynamic sampling (Yu et al., 2025), an asymmetric "clip-higher" surrogate (Yu et al., 2025), a constant-length-normalized objective (Liu et al., 2025), and a k3 KL anchor (Schulman, 2020) to the SFT init.

The headline result is that specialist RL helps when the trained specialist is the binding constraint of the pipeline, and helps very little otherwise. Thus training the extractor against a dense teacher-match reward improves held-out FinanceBench accuracy from 46 to 56 ($+10$) and QASPER token-F1 from 34.0 to 39.8 ($+6$), where a continuous ("smooth") reward is what unlocks the QASPER gain. On LongHealth the same recipe regresses (66 to 57). One would think a better extractor can only help the pipeline, so this looked at first like RL simply hurting, but it was in fact a producer/consumer coupling artifact. The synthesizer had been trained on the old extractor's output distribution, therefore re-SFT-ing it on the RL extractor's outputs ("co-adaptation") recovers the score to 67, and in the deployable frontier-orchestrator-and-synthesizer configuration the RL extractor reaches 74, within one point of a full frontier-extractor pipeline (75). I also find that joint end-to-end RL across all agents never beats plain SFT (38–42 vs 46), with per-agent credit beating shared credit but neither winning.

The contribution is a method (bottleneck-specialist RL with a smooth teacher-match reward, plus consumer co-adaptation) and a set of insights into its success and failure modes, namely objective alignment between the trained agent and the task metric, producer/consumer coupling in frozen pipelines, reward saturation, and the capacity ceiling of a 3B specialist. None of these required state-of-the-art performance to surface, and I think the failure modes are basically the most useful part of the study.

1 Introduction

Long-document question answering over financial filings, clinical records, and scientific papers is dominated by large frontier models. A cheaper alternative is a pipeline of small, specialized models. An orchestrator decomposes the question and routes document chunks to specialists (an extractor, a calculator, a reasoner), and a synthesizer composes the final answer. This MoSA-style design (Karabiyik et al., 2026) is attractive operationally, but it leaves a large accuracy gap to frontier models.

Supervised fine-tuning (SFT) on teacher traces gets the SLM pipeline part of the way, but SFT only imitates. The natural next step, and the subject of this project, is reinforcement learning. The question I care about is not only whether RL can push the specialists past their imitation ceiling, but where in the pipeline it actually pays off, since a multi-agent system gives many possible places to spend the RL budget. One would assume the synthesizer is the obvious target, because it is the agent that produces the answer, and yet the experiments below say the leverage is at the extractor that feeds it.

This is a CS224R project and the RL study is the novel contribution. The MoSA pipeline itself is carried over from a prior CS224N project (Karabiyik et al., 2026), with two changes I made for this work. First, chunk-based document dispatch, so specialists see roughly 3K-token chunks rather than whole 10–30K-token documents. Second, re-training the per-agent LoRA adapters (Hu et al., 2022) on the chunked traces, which fixes a train/inference distribution mismatch in the inherited adapters, since they had been trained on full documents but were being run on chunks. My contributions are:

- **RLOO+**, a from-scratch RLOO/GRPO-style trainer for the pipeline that composes several techniques from the recent LLM-RL literature, with two justified deviations (clip-higher and constant-length normalization).
- A **single-agent, bottleneck-first RL recipe** with a smooth teacher-match reward, and a **producer/consumer co-adaptation** step that re-SFTs a downstream agent on the RL agent's outputs.
- A **cross-benchmark analysis** on FinanceBench, QASPER, and LongHealth that maps when specialist RL helps, including several negative results and the mechanisms behind them.

2 Related Work

Policy-gradient RL and RLHF. The trainer builds on REINFORCE with a baseline for variance reduction (Williams, 1992; Sutton et al., 1999) and on the PPO clipped surrogate (Schulman et al., 2017). RLOO (Ahmadian et al., 2024) removes the value network and uses a leave-one-out group baseline, and GRPO (Shao et al., 2024) normalizes rewards within a group and folds the KL penalty into the loss. DAPO (Yu et al., 2025) contributes clip-higher and dynamic sampling, and Dr. GRPO (Liu et al., 2025) identifies and fixes a response-length bias in GRPO's per-sequence normalization. The KL anchor to a frozen reference follows RLHF practice (Christiano et al., 2017), and I use Schulman's k3 estimator (Schulman, 2020). DeepSeek-R1 (DeepSeek-AI, 2025) is the reference point for verifiable-reward RL at scale.

Multi-agent LLM systems. Mixture-of-Agents (Wang et al., 2024), Minions (Narayan et al., 2025), and Chain-of-Agents (Zhang et al., 2024) compose multiple LLM calls to improve quality or reduce cost, and MoSA is in this family. Prior work largely studies prompting and routing, thus RL fine-tuning of the individual specialists inside such a pipeline is comparatively underexplored, which is the gap this project targets.

Verifiable-reward RL. Recent reasoning-RL systems (Shao et al., 2024; Yu et al., 2025) use rule-based, verifiable rewards. My final-answer rewards (an LLM judge for FinanceBench, exact match for LongHealth, token-F1 for QASPER) are in this style. The per-agent teacher-match reward is my own design, and as far as I know rewarding a specialist by agreement with a teacher's sub-answer is not standard in this line of work.

3 Background and Setup

Pipeline. MoSA routes a question through a frozen orchestrator (DeepSeek-V3 over an API, or a local adapter) that emits sub-questions and chunk assignments, three specialists (extractor, calculator,

reasoner) that run on the assigned chunks, and a synthesizer that produces the final answer. Each agent is a Llama-3.2-3B model with a per-role QLoRA adapter (Dettmers et al., 2023) (NF4 4-bit, rank 16). All adapters are first distilled by SFT from DeepSeek-V3 teacher traces, and that SFT adapter is both the RL initialization and the frozen KL reference.

Benchmarks. FinanceBench (Islam et al., 2023) has numeric and textual answers over 10-K filings, scored by an LLM judge. LongHealth (Adams et al., 2024) is clinical multiple choice, scored by exact match over the A–E letter. QASPER (Dasigi et al., 2021) is scientific QA, scored by token-F1. I use the locked held-out splits from the prior project (FinanceBench 50, LongHealth 100, QASPER 100) and report single-sample evaluation unless noted.

4 Method

4.1 RLOO+

For each training query I sample a group of G completions from the policy and score each with a scalar reward. The advantage is the leave-one-out baseline

$$A_i = r_i - \frac{1}{G-1}\sum_{j \neq i} r_j,$$

so no value network is needed, which is basically the RLOO simplification of PPO (Ahmadian et al., 2024). Rewards are z-scored per task before forming advantages (Shao et al., 2024), and groups whose rewards have near-zero variance are dropped, since they carry no gradient. This dropping is the dynamic-sampling idea from DAPO (Yu et al., 2025), and it also avoids a 0/0 normalization on degenerate groups.

The policy loss is the PPO clipped surrogate (Schulman et al., 2017) on the response tokens,

$$\mathcal{L} = -\,\mathbb{E}\Big[\min\big(\rho A,\ \mathrm{clip}(\rho, 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}})\,A\big)\Big],$$

where $\rho = \pi_{\text{new}}/\pi_{\text{old}}$ is the importance ratio. I make two deviations from textbook PPO and GRPO, and I think both are justified rather than cosmetic. First, clip-higher. I decouple the bounds to $\varepsilon_{\text{low}} = 0.2$ below $\varepsilon_{\text{high}} = 0.28$ (Yu et al., 2025), so low-probability tokens can still gain mass. This implies the policy keeps some exploration, and of course that is what stopped the entropy collapse I saw with a symmetric clip. Second, constant-length normalization. I sum the per-token objective and divide by a constant $L_{\max}$ rather than by the per-sequence length (Liu et al., 2025), which removes the length bias that otherwise inflates long and often-wrong responses.

A k3 KL term to the frozen SFT reference (Schulman, 2020) is added to the loss with weight $\beta = 0.01$, and I keep the KL in the loss in GRPO style (Shao et al., 2024) rather than in the reward. The k3 form $\exp(s) - 1 - s$ with $s = \log(\pi_{\text{ref}}/\pi)$ is unbiased, low-variance, and always nonnegative, therefore the penalty behaves well. Hyperparameters are $G = 8$, learning rate 1e-6 (with one 5e-6 ablation), a cosine schedule with warmup, and a paged 8-bit AdamW optimizer. The loss, advantage, and KL are written from scratch, with unit tests checking that the advantages sum to zero within a group, that the KL is zero when policy equals reference, and that the loss reduces to $-\mathbb{E}[A]$ when the ratio is one.

4.2 Reward design

The final-answer rewards are verifiable and benchmark-specific. The per-agent teacher-match reward scores a specialist against the teacher's sub-answer, using relative error for numeric answers, token-F1 for short spans, and a local embedding similarity for free-form reasoning. I also use a smooth variant (continuous relative error and raw token-F1 instead of a thresholded 0/1), and this turned out to matter. A smooth reward gives gradient where a thresholded reward saturates, which is exactly the QASPER situation below.

4.3 Bottleneck-specialist RL and co-adaptation

Rather than train all agents jointly, I train the single specialist that is the binding constraint, which for these tasks is the extractor, and I keep the rest of the pipeline fixed. There is a catch. When a downstream agent has been SFT-trained on the old specialist's output distribution, RL-shifting the

specialist creates a distribution mismatch, so the frozen consumer can break even though the producer got better. I address this with co-adaptation. After RL-training the extractor, I re-SFT the synthesizer on traces produced by the RL extractor, thus adapting the consumer to the new producer. I also study joint end-to-end RL, training all specialists and the synthesizer against the shared final-answer reward, with two credit-assignment schemes, namely the shared final-answer advantage and per-agent advantages.

5 Experiments

5.1 Specialist RL helps where the specialist is the bottleneck

Table 1 reports held-out results, and Figure 1 shows the cross-benchmark picture. Training the extractor with a teacher-match reward improves FinanceBench from 46 to 56 (+10) and QASPER from 34.0 to 39.8 (+6). On QASPER the gain only appears with the smooth reward. A prior run with the thresholded reward left the score around 33, since the discrete reward gave each group almost no usable variance, therefore switching to the continuous reward is what moved the score to 39.8.

Table 1: Held-out accuracy (FinanceBench, LongHealth) and token-F1 (QASPER). LongHealth shows the raw RL extractor with the original synthesizer, then after co-adaptation.

Held-out	SFT	Extractor RL	Δ
FinanceBench (acc)	46	**56**	+10
QASPER (F1)	34.0	**39.8**	+6
LongHealth (acc)	66	57 raw $\rightarrow$ **67** co-adapt	recovered

5.2 LongHealth: a coupling regression, then recovery

On LongHealth the RL extractor with the original synthesizer regresses to 57. One would think a sharper extractor could only help downstream, so this looked at first like RL simply hurting, but it was in fact a producer/consumer coupling effect. The synthesizer was SFT-trained on the old extractor's output distribution, therefore when the extractor shifts the synthesizer is reading an input distribution it never saw. Co-adapting the synthesizer recovers the score monotonically, from 57 to 63 with a first co-adaptation pass, then to 67 with a cleaner and larger one (Figure 2), which is parity with SFT (66) within noise. In the deployable configuration with a frontier orchestrator and frontier synthesizer, the RL extractor reaches 74 versus 70 for the SFT extractor, thus landing within one point of a full frontier-extractor pipeline (75). The same frontier configuration on QASPER shows a wash-out instead, with RL at 46.9 and SFT at 48.1 (ceiling 48.4), within noise, because a strong synthesizer recovers the answer from any adequate chunks and the extractor's improvement no longer matters.

5.3 Joint end-to-end RL underperforms SFT

Training all agents jointly against the shared final reward never beats SFT on FinanceBench, landing at 38–42 versus 46 (Figure 3). One would assume training the whole pipeline end-to-end is strictly more powerful than training a single agent, and yet it is not. Per-agent credit (42) beats shared credit (38), which confirms that the shared final-answer advantage is too noisy a signal to attribute to any one agent, but neither variant wins. Together with the single-agent results, this is the main practical reason I concentrate RL on the bottleneck specialist rather than the whole pipeline.

5.4 Training and reward dynamics

This is a polishing regime, not a discovery one. Textbook RL expects a reward that starts low and climbs as the policy explores, with the policy moving meaningfully away from its initialization. I see the opposite. Because RLOO+ starts from a strong SFT init that already imitates the teacher, the reward starts high and moves only modestly, and the KL to the reference stays between about 1e-4 and 3e-4. On FinanceBench in particular the reward is nearly flat and the KL is essentially zero (Figure 4), and across the three benchmarks the mean reward shifts by at most about 0.14 over a full run (Figure 5). The policy is not discovering a behavior, it is basically being nudged around a

point that is already near a local optimum of the reward. This is the regime I would expect for RL fine-tuning on top of a competent imitator with a verifiable reward, and it is the opposite of R1-style training (DeepSeek-AI, 2025) where the policy travels a long way and long reasoning emerges.

What stalls learning is advantage variance, not reward magnitude. The RLOO+ advantage $A_i = r_i - \frac{1}{G-1}\sum_{j\neq i} r_j$ is built entirely from within-group reward variance. When the SFT model is already good, the G samples for a prompt tend to receive the same reward, so the variance is near zero, the dynamic-sampling filter drops the group, and there is no gradient at all. This implies that the better the initialization, the fewer informative groups there are, and therefore the effective learning signal shrinks as the policy approaches the teacher. The reward goes flat because it has been starved of advantages. QASPER is the clearest case. A thresholded reward (one if $F_1 \geq 0.5$, else zero) collapses a whole group to identical values, kills the variance, and learning flatlines around an F1 of 33. The smooth reward fixes this precisely because it restores variance, since a continuous F1 separates near-identical but unequal completions and manufactures the nonzero advantages the thresholded reward had destroyed. With the smooth reward the QASPER mean reward climbs from about 0.47 to 0.58 over training (Figure 5) and the score moves to 39.8. LongHealth is the mirror image and the most informative curve. One would assume a rising reward means a rising score, and yet there the mean reward also climbs, from about 0.39 to 0.53 (Figure 5), while the held-out score falls from 66 to 57. That is the cleanest single piece of evidence that the extractor's proxy reward improved while the pipeline metric got worse, which is exactly the coupling effect of Section 5.2. The lesson is a little counter-intuitive relative to from-scratch RL, where failure is common and variance is essentially free. Thus near a strong init, reward shaping (smoothness) matters more than reward scale, because the binding resource is advantage variance.

The tiny KL is a feature here. In RLHF one usually budgets KL to let the policy capture reward while guarding against drift and entropy collapse, and clip-higher (Yu et al., 2025) is the part of RLOO+ that protects that exploration. In this regime the policy barely needs to move, therefore the gains are small reweightings near the initialization rather than large behavioral change. That is of course why the runs are stable, and it is also why the headroom is limited, since a 3B policy that already matches the teacher cannot reweight its way past a capacity ceiling.

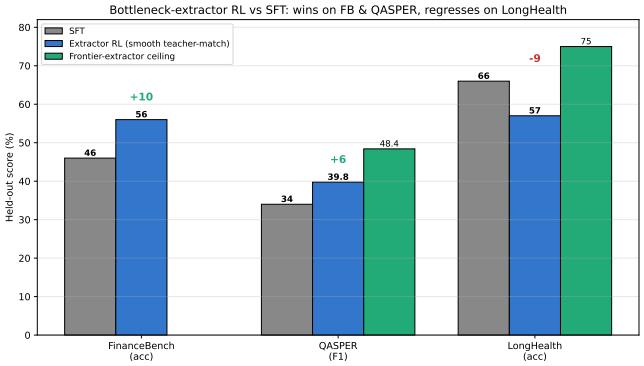

Figure 1: Extractor RL versus SFT across the three benchmarks. RL helps where extraction is close to the answer (FinanceBench, QASPER) and needs co-adaptation on LongHealth.

6 Discussion

The unifying picture is that specialist RL pays off when the trained agent is the binding constraint and its objective is aligned with the task metric. Three mechanisms explain the cross-benchmark pattern. First, objective alignment. When extraction is basically the answer, as with FinanceBench numbers and QASPER spans, improving the extractor improves the metric directly. On LongHealth the extractor produces evidence that a separate synthesizer turns into an A–E choice, therefore optimizing extraction-text-match is one step removed from the metric, and this is why naive RL there can hurt. Second, producer/consumer coupling. A frozen consumer trained on the old producer's distribution breaks when the producer shifts, and co-adaptation fixes it. I think this is a general recipe, namely RL a specialist, then re-SFT its consumer. Third, capacity and saturation ceilings. The frontier-extractor

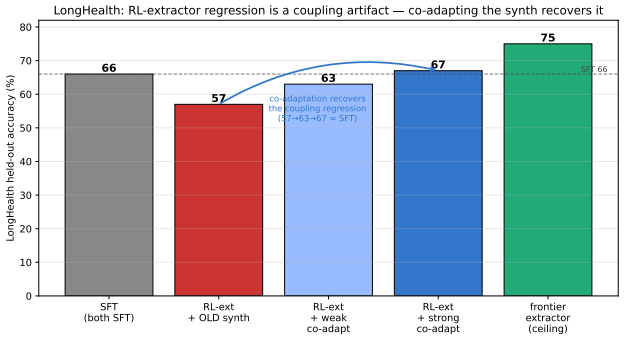

Figure 2: LongHealth: the RL extractor regresses with the frozen synthesizer, then recovers to parity and beyond as the synthesizer is co-adapted to the RL extractor's outputs.

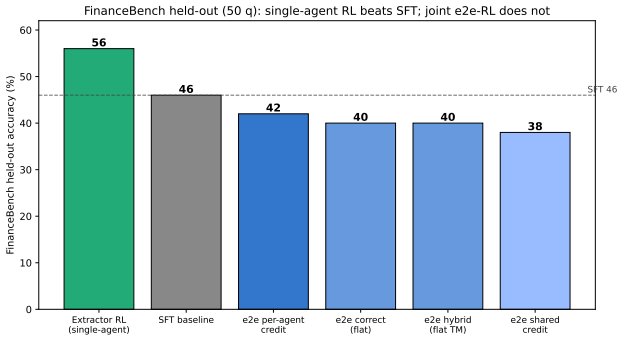

Figure 3: Joint end-to-end RL on FinanceBench stays below the SFT baseline, with per-agent credit above shared credit.

ceilings (LongHealth 75, QASPER 48.4) sit well above the 3B, and on QASPER the teacher-match reward is saturated, since the SFT extractor already matches the teacher, so further gains there need a task-aligned reward or more capacity rather than more steps.

A taxonomy of outcomes. Putting the dynamics and the cross-benchmark results together, the four behaviors I see map cleanly onto when group-relative RL from a strong init should and should not work. The policy *improves* when the proxy reward is aligned with the metric, the trained agent is the bottleneck, and there is enough reward variance to form advantages (FinanceBench). It *stalls* when the reward saturates and the within-group variance dies, which calls for shaping or more capacity rather than more steps (QASPER teacher-match). It *regresses* when the agent's own reward improves but it shifts the input distribution of a frozen downstream consumer, which is basically the same failure as evaluating a fixed policy off its training distribution in offline RL, therefore the fix is to adapt the consumer rather than to train the producer harder (LongHealth). And it *washes out* when a strong downstream makes the agent's improvement irrelevant to the metric (QASPER under a frontier synthesizer). None of these are visible from the final scores alone, which is why I think the reward dynamics are the more useful half of the study.

7 Limitations and Future Work

The trained specialists are 3B models, and the frontier ceilings show a real capacity gap that RL cannot close. The QASPER result is reward-limited rather than training-limited, therefore a task-aligned reward (rewarding the extractor by the final answer rather than by teacher-match) is the natural next step, but it washes out under a frontier synthesizer, which means its upside is mainly the all-SLM regime. LongHealth's full-context extractor RL is memory-bound on a single GPU. Finally, the evaluations use single-sample scoring on locked held-out splits, so a majority-vote protocol would likely raise all of the numbers and is worth running before drawing sharp comparisons.

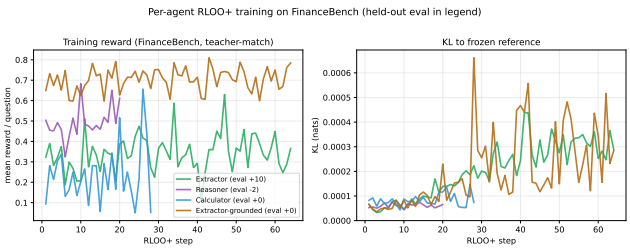

Figure 4: FinanceBench training dynamics. The teacher-match reward edges up while the KL to the SFT reference stays tiny, so the policy improves without drifting far from its initialization.

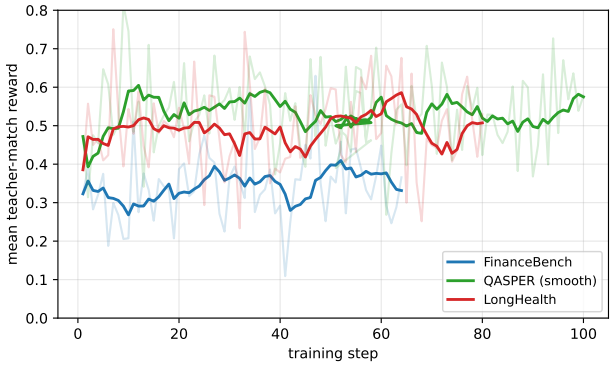

Figure 5: Mean teacher-match reward over training for the three single-agent extractor runs (rolling average). QASPER and LongHealth climb the most, but on LongHealth the rising reward coincides with a drop in held-out accuracy, which is the coupling effect of Section 5.2.

8 Conclusion

Specialist RL is a useful but conditional lever for small-model multi-agent QA. With a from-scratch RLOO+ trainer, a smooth teacher-match reward, and producer/consumer co-adaptation, bottleneck-specialist RL beats SFT on FinanceBench ($+10$) and QASPER ($+6$) and reaches near-frontier extraction quality on LongHealth in the deployable configuration, while joint end-to-end RL does not beat SFT. The takeaway I would keep is a small map of when the lever works. Align the trained agent's objective with the task metric, target the binding constraint, and co-adapt its consumer.

Team Contributions

This is a solo project. All work was carried out by Mert Karabiyik. The MoSA pipeline is inherited from a prior CS224N project, and the chunk-based dispatch, the LoRA re-training, and all RL work in this report are new for CS224R. This matches the contribution breakdown in the proposal, with the one scope change that RL moved from the synthesizer to the bottleneck extractor once the experiments showed the extractor was the binding constraint.

AI Tools Disclosure

I used the AI assistant Claude as an aid throughout the project, and not as a substitute for the core work. The reinforcement learning algorithm itself, namely the RLOO+ loss, advantage, and KL in `rl/losses.py` and the trainer in `rl/train_agent.py`, I developed completely independently. AI helped me with boilerplate code, setting up the evaluation harness, and data management such as chunk routing and synthetic data generation, and it also aided me in tracking and coordinating training runs on Modal, debugging, and pulling insights out of the pipeline traces. It also helped me generate figures and draft and organize the final report and its references, which I then revised,

fact-checked against my own results, and rewrote in my own voice. Everything the tools produced was reviewed and edited by me before it went into the submission, therefore the judgments and the final wording are my own.

References

Lisa Adams, Felix Busch, Tianyu Han, Jean-Baptiste Excoffier, Matthieu Ortala, Alexander Löser, Hugo J. W. L. Aerts, Jakob Nikolas Kather, Daniel Truhn, and Keno Bressem. 2024. LongHealth: A Question Answering Benchmark with Long Clinical Documents. *arXiv preprint arXiv:2401.14490* (2024).

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. *arXiv preprint arXiv:2402.14740* (2024).

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers. In *NAACL*.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948* (2025).

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. In *Advances in Neural Information Processing Systems*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. FinanceBench: A New Benchmark for Financial Question Answering. *arXiv preprint arXiv:2311.11944* (2023).

Mert Karabiyik, Haseeb Ismail, and Shayaan Memon. 2026. MoSA: Mixture-of-Specialized-Agents for Cost-Efficient Long-Document Question Answering. *CS 224n Custom Project, Stanford University* (2026).

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding R1-Zero-Like Training: A Critical Perspective. *arXiv preprint arXiv:2503.20783* (2025).

Avanika Narayan, Dan Biderman, Sabri Eyuboglu, Avner May, Scott Linderman, James Zou, and Christopher Ré. 2025. Minions: Cost-efficient Collaboration Between On-device and Cloud Language Models. *arXiv preprint arXiv:2502.15964* (2025).

John Schulman. 2020. Approximating KL Divergence. `http://joschu.net/blog/kl-approx.html`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300* (2024).

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2024. Mixture-of-Agents Enhances Large Language Model Capabilities. *arXiv preprint arXiv:2406.04692* (2024).

Ronald J Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning* 8 (1992), 229–256.

Qiying Yu et al. 2025. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv preprint arXiv:2503.14476* (2025).

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arik. 2024. Chain of Agents: Large Language Models Collaborating on Long-Context Tasks. In *Advances in Neural Information Processing Systems*.