ToolBrain: A Flexible Reinforcement Learning Framework for Agentic Tools

Demonstration Track

Quy Minh Le*
ToolBrain Research
Dublin, Ireland

Minh Sao Khue Luu*
ToolBrain Research
Dublin, Ireland

Khanh-Tung Tran*
University College Cork
Cork, Ireland

Duc-Hai Nguyen
University College Cork
Cork, Ireland

Hoang-Quoc-Viet Pham
ToolBrain Research
Dublin, Ireland

Quan Le
CeADAR, University College Dublin
Dublin, Ireland

Hoang Thanh Lam†
IBM Research Lab
Dublin, Ireland

Hoang D. Nguyen†
University College Cork
Cork, Ireland

ABSTRACT

Training language-enabled agents to reliably use tools remains a significant challenge, often hindered by complex frameworks and high computational costs. We present ToolBrain, an open-source platform that addresses this challenge through the Coach–Athlete paradigm, an architectural abstraction that simplifies the application of reinforcement learning (RL) to agentic workflows. We evaluate ToolBrain by adapting a compact language model to three representative tasks: multi-step information retrieval, quantitative reasoning, and real-world API interaction. Our results demonstrate that ToolBrain substantially improves the performance of compact models, enabling them to solve complex tool-using tasks efficiently. The accompanying demonstration video can be viewed at https://youtu.be/FIgfg-y0sXw

KEYWORDS

Agent-Based Systems; Language-Enabled Agents; Reinforcement Learning; Tool Use; Large Language Models (LLMs); Open-Source Framework

ACM Reference Format:
Quy Minh Le, Minh Sao Khue Luu, Khanh-Tung Tran, Duc-Hai Nguyen, Hoang-Quoc-Viet Pham, Quan Le, Hoang Thanh Lam, and Hoang D. Nguyen. 2026. ToolBrain: A Flexible Reinforcement Learning Framework for Agentic Tools: Demonstration Track. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 3 pages. https://doi.org/10.65109/ZKRA7271

1 INTRODUCTION

The advent of large language models (LLMs) has spurred the development of *language-enabled agents* capable of complex behaviors such as planning, reasoning, and interacting with external tools [18, 20, 22]. However, training these agents to use tools reliably remains a significant challenge. Many existing systems rely on prompt engineering or supervised fine-tuning, which limits their ability to learn from experience. Although reinforcement learning (RL) provides a powerful paradigm for adaptive learning, its integration into agentic workflows is often limited by complex frameworks, high computational costs, and the difficulty of reward design [6, 12].

ToolBrain is an open-source platform designed to address these challenges by making RL-based training for tool-using agents more accessible through a unified API that abstracts away the complexities of the training loop. To position our contributions within the existing landscape, we compare ToolBrain with several prominent frameworks in Table 1.

Table 1: Comparison of ToolBrain with other frameworks.

Aspect	ToolBrain	LangChain / LangGraph	ART	Agent Lightning
Training Approach	✓ **Native RL (GRPO, DPO)** with iterative fine-tuning [16, 19].	Supervised learning & prompt chaining [11].	GRPO-based RL with RULER evaluator [6].	Hierarchical RL with credit assignment [12].
Reward System	✓ **Hybrid:** Python callable + **ranking-based** LLM [14, 23].	Manual heuristic scoring.	RULER: automated LLM-as-judge with relative scoring.	Credit-aware reward assignment.
Tool Management	✓ **Integrated Tool Retriever** automatically selects relevant tools, a strategy explored in prior work such as [9, 15, 18].	Manual tool definition and passing.	Manual tool definition and passing.	Manual tool definition and passing.
Advanced Strategies	✓ Supports **Knowledge Distillation** [7, 17] and **Zero-Learn** task generation [18, 21].	–	–	–
Efficiency & Usability	✓ **Simple `Brain` API**; integrated **Unsloth/QLoRA** optimizations [2, 5, 8, 13].	Code-centric; complex context management.	Minimal code changes; requires separate server setup.	Requires MDP design; steep RL expertise.

2 THE COACH-ATHLETE PARADIGM

ToolBrain's architecture is structured around the **Coach–Athlete paradigm**, a conceptual abstraction that cleanly separates *training*

*Contributed equally to this work.
†Corresponding authors: t.l.hoang@ie.ibm.com, hn@cs.ucc.ie

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/ZKRA7271

orchestration from *task execution*. The novelty of ToolBrain lies in its architectural abstraction that decouples training from execution and enables RL integration into heterogeneous agent frameworks. As illustrated in Figure 1, it consists of three key components.

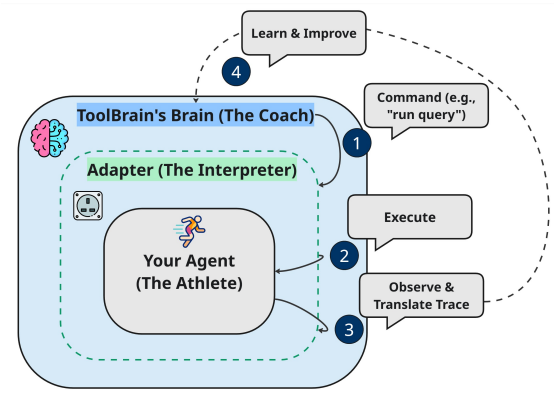

Figure 1: The Coach-Athlete paradigm. The Brain (Coach) orchestrates the learning loop: it observes the Agent's (Athlete) actions via the Adapter and then updates the Agent's underlying policy to improve performance.

The **Brain** (the Coach) is the high-level API that manages the entire training loop. The user-provided **Agent** (the Athlete) is responsible for task execution and is unaware of the training process. The crucial link is the internal **Adapter** (the Interpreter). Implementing the classic Adapter design pattern [4], it acts as a translation layer that wraps the user's agent. Its core function is to convert the heterogeneous, framework-specific memory of the Agent into a standardized, high-fidelity execution trace. The Brain then uses this trace to internally perform the *Learn & Improve* step, computing the required policy updates. Finally, it applies these updates to the Agent's underlying language model, completing the learning cycle. Although this paradigm shares structural similarities with traditional actor–learner architectures such as IMPALA [3], the separation of concerns is explicitly designed to support the iterative development cycle of tool-using agents. As a result, the workflow becomes modular, allowing strategy and reward changes via the Brain API without altering the core agent logic or structure.

3 SYSTEM DEMONSTRATION

Our demonstration shows how compact models adapt to diverse tool-use tasks while reusing a unified training pipeline across heterogeneous agent architectures and domains.

- **Email Search Agent:** Navigates a large email corpus using search and read tools to resolve multi-step information retrieval queries over unstructured data.
- **Finance Agent:** Maps natural language queries to financial calculation tools for structured quantitative reasoning tasks.
- **API Agent:** Calls an external weather API to answer real-time queries with grounded and up-to-date information.

4 EXPERIMENTAL RESULTS

We evaluated ToolBrain's effectiveness on the three tasks described above. The main experiment on the Email Search Agent provides an analysis of learning dynamics, while the supplementary experiments highlight the framework's rapid adaptation capabilities.

4.1 Main Experiment: Email Search Agent

We trained two sizes of the Qwen2.5 model (3B and 7B parameters) for 60 steps using GRPO [19] with an LLM-as-a-Judge [14, 23]. The task required the agent to answer questions from a benchmark dataset [1] by navigating the Enron email corpus [10]. The results of a representative run are summarized in Table 2. Both models show significant improvements over their initial baselines. Notably, after training, the 7B model's task success rate more than triples, and its hallucination rate is nearly halved. This demonstrates that ToolBrain can effectively teach complex, multi-step tool use. While commercial models like GPT-4o-mini offer strong zero-shot performance, fine-tuned compact local models (3B/7B) provide advantages in privacy, latency, and deployment cost for domain-specific tasks.

Table 2: Evaluation of the Email Search Agent, comparing performance before (Step 0) and after (Step 60) training. ↓ indicates lower is better.

	Before Training (Step 0)			After Training (Step 60)		
Model	Success Rate%	Hallucination%↓	Turns↓	Success Rate%	Hallucination%↓	Turns↓
Qwen2.5-3B	0.0	100.0	4.63	**16.7**	66.7	5.57
Qwen2.5-7B	13.3	60.0	7.03	**43.3**	35.0	4.77

4.2 Supplementary Experiments: Flexibility

To further examine the framework's adaptability, we evaluated a 0.5B parameter agent on the Finance and API tasks. For each task, a training set of 40 queries and a test set of 10 queries were synthesized via the framework's Zero-Learn task generation mechanism. The agents were then trained for a small number of optimization steps using only knowledge distillation [7, 17]. As shown in Table 3, this lightweight training procedure resulted in a twofold increase in task success rates for both agents. These results indicate that ToolBrain provides an efficient mechanism for adapting compact models to new domains with minimal training overhead.

Table 3: Flexibility on Secondary Tasks. Success Rate (%) is measured on a separate test set of 10 queries.

Case Study	Untrained	Trained (w/ Distill)
Finance (Quantitative Reasoning)	20.0%	40.0%
API (Real-World Grounding)	30.0%	60.0%

5 CONCLUSION

We presented ToolBrain, an open-source framework featuring the Coach-Athlete paradigm that simplifies the training of language-enabled agents. Our demonstration and results validate its effectiveness in enhancing agent capabilities across multiple tasks. The source code is available at https://github.com/toolbrain/toolbrain.

ACKNOWLEDGMENTS

This publication has emanated from research supported in part by grants from Research Ireland under Grant [12-RC-2289-P2] and [18/CRT/6223] which is co-funded under the European Regional Development Fund. The authors also acknowledge ISY Labs for providing the computational resources used in this work. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

REFERENCES

[1] Kyle Corbitt. 2025. Enron Emails Sample Questions. Hugging Face Datasets. https://huggingface.co/datasets/corbt/enron_emails_sample_questions

[2] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. arXiv:2305.14314 [cs.LG] https://arxiv.org/abs/2305.14314

[3] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. 2018. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. arXiv:1802.01561 [cs.LG] https://arxiv.org/abs/1802.01561

[4] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design patterns: elements of reusable object-oriented software.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, United States. 395 pages.

[5] Daniel Han and Michael Han. 2023. Unsloth. https://github.com/unslothai/unsloth. GitHub repository.

[6] Brad Hilton, Kyle Corbitt, David Corbitt, Saumya Gandhi, Angky William, Bohdan Kovalenskyi, and Andie Jones. 2025. ART: Agent Reinforcement Trainer. https://github.com/openpipe/art.

[7] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531 [stat.ML] https://arxiv.org/abs/1503.02531

[8] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL] https://arxiv.org/abs/2106.09685

[9] Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Gavin Li, Junze Zhang, Di Yin, Shruti Marwaha, Jennefer N. Carter, Xin Zhou, Matthew Wheeler, Jonathan A. Bernstein, Mengdi Wang, Peng He, Jingtian Zhou, Michael Snyder, Le Cong, Aviv Regev, and Jure Leskovec. 2025. Biomni: A General-Purpose Biomedical AI Agent. bioRxiv preprint. https://doi.org/10.1101/2025.05.30.656746

[10] Bryan Klimt and Yiming Yang. 2004. The Enron Corpus: A New Dataset for Email Classification Research. In *Machine Learning: ECML 2004 (Lecture Notes in Computer Science, Vol. 3201)*, Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 217–226. https://doi.org/10.1007/978-3-540-30115-8_22

[11] Inc. LangChain. 2025. LangGraph. https://www.langchain.com/langgraph.

[12] Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. 2025. Agent Lightning: Train ANY AI Agents with Reinforcement Learning. arXiv:2508.03680 [cs.AI] https://arxiv.org/abs/2508.03680

[13] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed Precision Training. arXiv:1710.03740 [cs.AI] https://arxiv.org/abs/1710.03740

[14] 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. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL] https://arxiv.org/abs/2203.02155

[15] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. arXiv:2307.16789 [cs.AI] https://arxiv.org/abs/2307.16789

[16] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. https://doi.org/10.48550/arXiv.2305.18290 arXiv:2305.18290 [cs].

[17] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108 [cs.CL] https://arxiv.org/abs/1910.01108

[18] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv:2302.04761 [cs.CL] https://arxiv.org/abs/2302.04761

[19] 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. https://doi.org/10.48550/arXiv.2402.03300 arXiv:2402.03300 [cs].

[20] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. arXiv:2303.11366 [cs.AI] https://arxiv.org/abs/2303.11366

[21] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. arXiv:2212.10560 [cs.CL] https://arxiv.org/abs/2212.10560

[22] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629 [cs.CL] https://arxiv.org/abs/2210.03629

[23] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv:2306.05685 [cs.CL] https://arxiv.org/abs/2306.05685