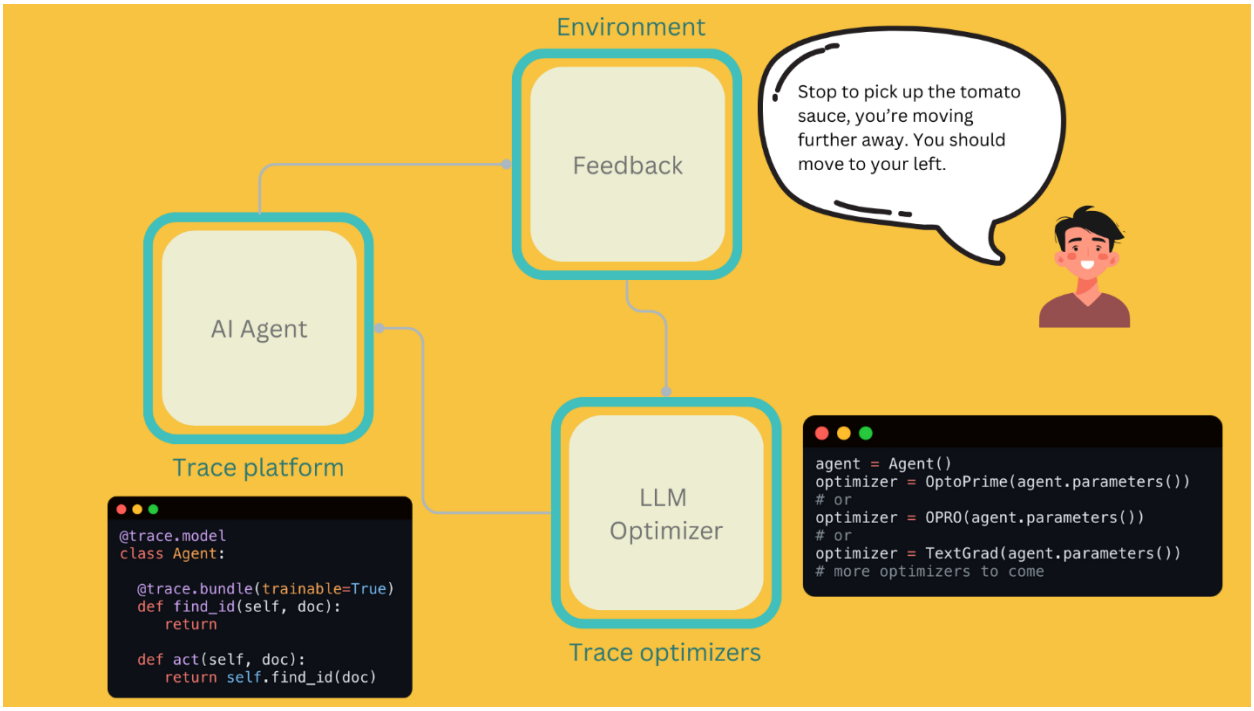
Trace

Microsoft Research

January 2025

Introduction

Trace is an open-source, PyTorch-like library for training AI agents using feedback (like scores, natural language, errors, etc.). Trace and Generative Optimization enable a new kind of deep learning that goes beyond tensor-based models. With Trace, any Python method or class can be viewed as model and we can easily train Python objects (like prompts and hyperparameters) and codes in-place end-to-end, just like training neural networks -- but with more intuitive and more interpretable models.



New Release Highlight

0.1.3. Trace now supports multiple LLM-based optimizers including OptoPrime, TextGrad, OPRO, and optimizing asyncio workflows.

Key Features

End-to-End Generative Optimization: An AI agent has many modules. Trace captures the system's underlying execution flow and represents it as a graph. Trace can then optimize the entire system with general feedback using LLM-based optimizers.

Native Python Support: Trace gives users full flexibility in programming AI agents. By using two primitives (node and bundle) to wrap over Python objects and functions, Trace is compatible with any Python program and capable of optimizing any mixture of code, string, numbers, objects, etc.

Research Platform: Trace propagates execution traces as *Minimal Subgraphs* to optimizers. This common abstraction allows easy experimentation with different optimization algorithms (e.g., OPRO, TextGrad, OptoPrime) in Trace and gives researchers full freedom to design new techniques for AI agents.

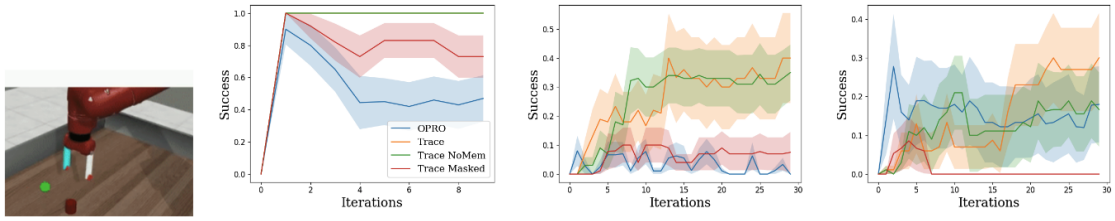
Benchmark Results

0-shot	BBH all (23 tasks)	NLP (12 tasks)	Algorithmic (11 tasks)	0-shot	BBH all (23 tasks)	NLP (12 tasks)	Algorithmic (11 tasks)
DSPy	41.6	53.8	32.6	DSPy + CoT	70.4	73.7	68.0
DSPy-PO	55.3	69.0	45.2	DSPy-PO + CoT	71.6	73.9	70.0
Trace	**59.5**	**70.9**	**51.1**	Trace + CoT	**78.6**	**75.8**	**80.6**

Table 1: End-to-end workflow optimization for an LLM benchmark (Big-Bench Hard) in 0-shot setup. `CoT` refers to Chain-of-Thought prompting and `PO` refers to DSPy's own prompt optimizer (COPRO). We use Trace to optimize a DSPy program, starting from the same program and prompt template specified by DSPy.

	OptoPrime (Trace)	Time	TextGrad (24-10-30)	Time	TextGrad (Trace)	TextGrad (Reported)
MMLU-Machine Learning	**86.6** (0.2)	1.7 (0.6)	86.1 (0.5)	3.5 (1.1)	86.3 (0.2)	88.4
MMLU-College Physics	**94.1** (0.8)	1.2 (0.3)	93.1 (0.7)	2.3 (0.4)	93.3 (0.6)	95.1
Google-proof QA	**59.6** (1.3)	12.2 (1.4)	53.2 (0.6)	19.5 (1.9)	54.0 (0.7)	55.0
BBH Counting	**89.4** (0.1)	55.9 (4.5)	89.2 (1.2)	142.9 (9.3)	87.6 (1.7)	91.9
BBH Word Sorting	71.6 (3.1)	82.5 (10.1)	**72.0** (0.4)	211.1 (16.8)	71.4 (2.5)	79.8
GSM8K	**82.5** (0.1)	—	82.4 (0.6)	—	82.0 (0.2)	81.1

Table 2: Comparison between Trace and TextGrad. The optimizer is GPT-4o-2024-08-06, and the student model is GPT-35-turbo-1106. The results show the mean and the standard error of success rate of the last iterate computed by 5 seeds. The experiment time reported is in minutes (the time involves not just training but also validation and testing by running TextGrad's original pipeline); the time of GSM8K experiment is omitted as the experiment time (>8hrs) is determined primarily by the evaluation not optimization.



(a) Reach (Test) **(b)** Pick-place (Test) **(c)** Push (Test)

Figure 6: Learning the feedback control policy (code) for a simulated Sawyer manipulator in LLF-Bench Meta-World. In each iteration (x-axis), one episode of rollout (10 steps) is performed, and then the policy is updated. The mean and standard error of the success rate over 10 seeds are shown.

Scenarios Where This Tech Shines

Trace is designed as a research platform to design the next generation agents that can self-adapt their codes, prompts, parameters based on feedback in interactions. Near-term usages include prompt + code optimization of agentic systems and learning through interaction to write super-expert code (for cases where engineering is hard) (see e.g. paper).

Contact Us

Project contact: AIF-Trace@microsoft.com