Trace is the New AutoDiff — Unlocking Efficient Optimization of Computational Workflows

Ching-An Cheng[*]
Microsoft Research
chinganc@microsoft.com

Allen Nie[*]
Stanford University
anie@stanford.edu

Adith Swaminathan[*]
Microsoft Research
adswamin@microsoft.com

Abstract

We study a class of optimization problems motivated by automating the design and update of AI systems like coding assistants, robots, and copilots. We propose an end-to-end optimization framework, Trace, which treats the computational workflow of an AI system as a graph akin to neural networks, based on a generalization of back-propagation. Optimization of computational workflows often involves rich feedback (e.g. console output or user's responses), heterogeneous parameters (e.g. prompts, hyper-parameters, codes), and intricate objectives (beyond maximizing a score). Moreover, its computation graph can change dynamically with the inputs and parameters. We frame a new mathematical setup of iterative optimization, Optimization with Trace Oracle (OPTO), to capture and abstract these properties so as to design optimizers that work across many domains. In OPTO, an optimizer receives an execution trace along with feedback on the computed output. Trace is the tool to implement OPTO in practice: Trace has a Python interface that efficiently converts a computational workflow into an OPTO instance using a PyTorch-like interface. Using Trace, we develop a general-purpose LLM-based optimizer called OptoPrime that can effectively solve OPTO problems. In empirical studies, we find that OptoPrime is capable of first-order numerical optimization, prompt optimization, hyper-parameter tuning, robot controller design, code debugging, etc., and is often competitive with specialized optimizers for each domain. We believe that Trace, OptoPrime and the OPTO framework will enable the next generation of interactive agents that automatically adapt using various kinds of feedback. Website: https://microsoft.github.io/Trace/.

1 Introduction

Computational workflows that integrate large language models (LLMs), machine learning (ML) models, orchestration, retrievers, tools, etc., power many state-of-the-art AI applications [1]: from chatbots [2], coding assistants [3], robots [4], to multi-agent systems [5]. However designing a computational workflow requires laborious engineering because many heterogeneous parameters (e.g. prompts, orchestration code, and ML hyper-parameters) are involved. Moreover, after deployment any erroneous behaviors of the workflow persist unless a developer manually updates it.

We study a class of optimization problems motivated by automating the design and update of computational workflows. Computational workflows produce optimization problems with heterogeneous parameters, rich feedback (e.g. console output and user's verbal responses), and intricate objectives (beyond maximizing a score). Moreover, a workflow can have interdependent steps (e.g. adaptive orchestration, feedback control loops) and/or involve semi-black-box operations whose behavior cannot be succinctly captured(e.g. ML models, simulations). As a result, the structure of the computation may change as the parameters and the inputs of the workflow vary.

[*]Equal contribution

Preprint. Under review.