AutoTuneX: Interactive Automated Fine-Tuning for Large Language Models

Daniel Karl I. Weidele, Priyanshu Rai, Frederico Araujo, Teryl Taylor, Radu Marinescu

IBM Research

{daniel.karl, priyanshu.rai, frederico.araujo, terylt}@ibm.com, radu.marinescu@ie.ibm.com

Abstract

We present AutoTuneX, a system architecture design and implementation for users to interactively fine-tune large language models (LLMs) based on automated hyperparameter optimization particularly built around Bandit Limited Discrepancy Search. Next to a classical Graphical User Interface (GUI) our system features an agentic runtime to facilitate automated fine-tuning via chat.

Demo video — https://youtu.be/rx7vz-ws7vk

Introduction

Large language models' (LLMs) increasing success in machine learning (ML) finds real-world application in software products of various domains. To further enhance the accuracy of artificial intelligence (AI) across ML tasks, AI engineers resort to a process called *fine-tuning*. In fine-tuning, a pre-trained LLM is presented with select input-output training data pairs, for it to adjust some of its internal weights to better meet the target outputs when presented with a given task. Fine-tuning is subject to a larger set of hyperparameters: slices of input-output pairs, tuning methodologies with their exact parameters or loss functions need to be selected.

In this work we present AutoTuneX, a full-stack system design and implementation towards the automation of hyperparameter optimization for LLM fine-tuning. AutoTuneX comprises of a scalable backend, as well as an intuitive and interactive graphical user interface to enable beginners and experts alike. We further present an agentic solution to conversationally drive the process of automated fine-tuning.

Related Work. Our proposed system is related to Hugging Face's AutoTrain (Thakur 2024), a no-code platform to train ML models including LLMs. However, unlike AutoTuneX, AutoTrain lacks hyperparameter optimization thus limiting automation, requiring users to manually configure all tunable parameters, and does not offer agentic operation.

AutoTune: Automated Fine-Tuning of LLMs

At the core of AutoTuneX lies *AutoTune*–our distributed algorithm that explores the search space defined by hyperparameter bands for various fine-tuning strategies to optimize model performance. AutoTune's default search strategy is based on the recently introduced Bandit Limited Discrepancy Search (BLDS) (Kishimoto et al. 2022), which we extend to support LLMs in addition to classical ML models. Specifically, BLDS is a guided depth-first search method that starts from a default configuration of hyperparamters and gradually explores nearby alternatives. Starting with one allowed change–called discrepancy–from an initial setup, BLDS incrementally increases this limit, focusing the search around promising defaults rather than exploring at random.

AutoTune is implemented in Ray Tune (Liaw et al. 2018) to enable distributed training across multiple GPUs. We support full supervised fine-tuning (SFT) and parameter-efficient fine-tuning (PEFT) techniques, such as prompt-tuning (Lester, Al-Rfou, and Constant 2021) or LoRA (Hu et al. 2021), or even variations such as Activated LoRA (aLoRA) (Greenewald et al. 2025), a recent extension of LoRA that enables efficient reuse of model's key-value (KV) cache. In addition to BLDS, AutoTune includes distributed implementations of popular Bayesian optimization search methods such as Hyperopt (Bergstra, Yamins, and Cox 2013) and Hyperband (Li et al. 2016), as well as random search. However, recent work by (Tomar et al. 2025) showed that BLDS typically outperforms the latter search strategies for hyperparamter optimization.

AutoTuneX: Interactive Experience for AutoTune

AutoTuneX wraps the *AutoTune* algorithm in a service layer and exposes it along with tangential functionality as a REST API For example, we hook into Ray execution at various points from the service layer, to gather statistics and logging information that can be saved in an SQL database. We containerize our platform as a Docker image to deploy and run in cloud. Figure 2 shows a schematic overview of the full AutoTuneX stack including its web-based user experience.

Graphical User Interface. Our Svelte-based graphical user interface (GUI) follows previous design for optimization work (Weidele et al. 2023; Franke et al. 2024) and communicates with the REST API directly from the client's browser. To design a new tuning experiment from scratch, users would first upload their input-output pair data set using a data upload wizard. Optionally for more experienced

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

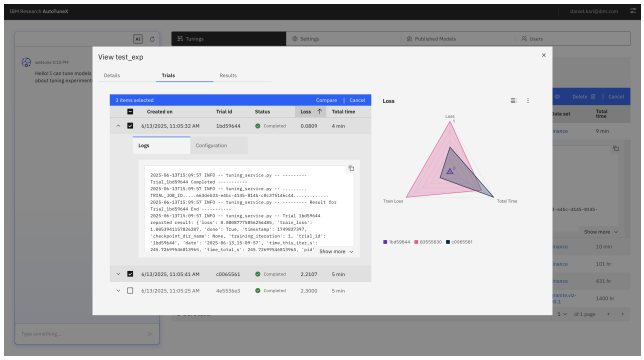

Figure 1: Results tab in AutoTuneX GUI: metrics of different LLM fine-tuning trials can be compared graphically (right). Users can explore hyperparameter configuration and logs of the trials in a leaderboard (left).

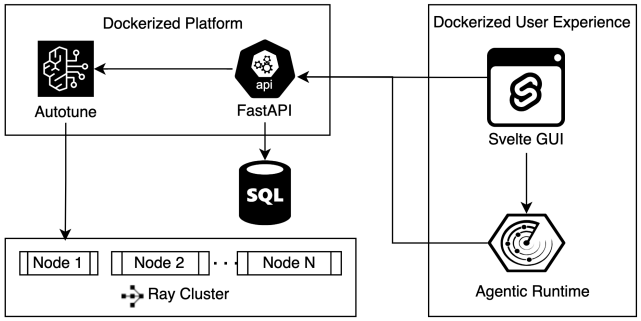

Figure 2: Architecture overview: Hyperparameter optimization scales with available number of nodes in a Ray cluster. Our FastAPI service layer orchestrates the execution and keeps track of configurations, data sets, logs and metadata in an SQL database, providing support for a classical GUI and a conversational agentic tool-calling runtime.

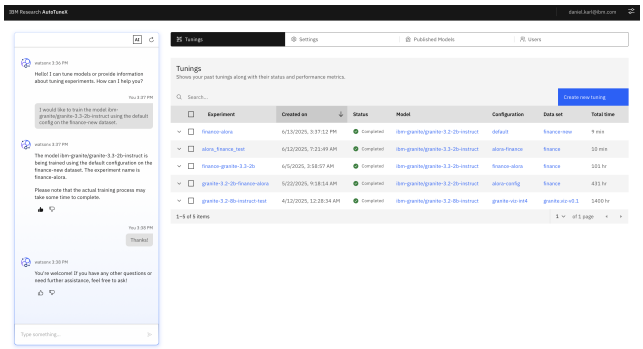

Figure 3: Conversational Agent in AutoTuneX: the agent controls the underlying REST API to dynamically execute user requests. Equipped with GET and POST tools, the agent can further plan and execute a sequence of API calls. Lastly, the runtime allows for the user to be looped back in, for the agent to circle back with the user for confirmation.

users, the GUI allows for the creation of new optimization configurations to adjust hyperparameter search bands and resource allocations (i.e. number of GPUs, maximum RAM, etc.). The user then selects their desired combination of model, configuration and data set to start an execution. After successful completion, and even in real-time during the execution, users can observe the trial leaderboard and visually assess evaluation metrics via a Radar chart (ref. figure 1), or select trials for detailed comparison using a tabular view, which highlights differences and similarities of hyperparameter configurations. After completion, users can download the best trial's model weights as a .zip file, or push the weights to a proprietary model registry, from where the model can further be hosted for inference. The .zip file further contains an installation and execution scripts to make it especially easy for non-expert users to run locally (if the model size allows).

Conversational Agentic Automation. In our custom, plugin-centric agentic framework, we further implemented a conversational agent to facilitate *fully automated fine-tuning via chat* (ref. figure 3). Next to authoring agents as no-code YAML files or custom Python modules, our runtime facilitates tool-calling and allows for integration of emerging agentic patterns and other open-source frameworks. This modular design allows us to operate AutoTuneX capabilities using a no-code approach through a Reasoning and Acting (ReAct) (Yao et al. 2022) agent. As for tools, we expose `fetch_swagger_docs` to directly retrieve and parse the AutoTuneX OpenAPI REST specifications. This enables the agent to autonomously select relevant endpoints based on user queries. To execute API calls we further include tools `http_get_client` and `http_post_client`. By few-shot prompt engineering we teach various user scenarios, allowing the agent to more effectively map natural language requests to specific (chains of) API operations.

As a result, our agent intelligently interprets user queries, automatically selects and plans appropriate API endpoints, and then executes GET and POST requests as needed. This solution enables users to interact with the platform through natural language instructions in a conversational way, performing tasks such as retrieving job status or configuration details, and exploring trial configurations, while receiving clear, summarized responses.

Conclusion and Future Work

We presented AutoTuneX, a full-stack system design and implementation for automated fine-tuning of LLMs. Based on our scalable AutoTune algorithm exposed via REST API, we demonstrate support for two modes of interaction: GUI and conversational agent. In the next step we plan to execute usability studies in form of A/B studies with experienced and novice users alike, to evaluate both agent and GUI-only mode. We hope for our software stack to level the playing field to further democratization of AI.

References

Bergstra, J.; Yamins, D.; and Cox, D. 2013. Making a Science of Model Search: Hyperparameter Optimization

in Hundreds of Dimensions for Vision Architectures. In Dasgupta, S.; and McAllester, D., eds., *Proceedings of the 30th International Conference on Machine Learning*, volume 28(1) of *Proceedings of Machine Learning Research*, 115–123. Atlanta, Georgia, USA: PMLR.

Franke, L.; Weidele, D. K. I.; Dehmamy, N.; Ning, L.; and Haehn, D. 2024. AutoRL X: Automated Reinforcement Learning on the Web. *ACM Transactions on Interactive Intelligent Systems*, 14(4): 1–30.

Greenewald, K.; Lastras, L.; Parnell, T.; Shah, V.; Popa, L.; Zizzo, G.; Gunasekara, C.; Rawat, A.; and Cox, D. 2025. Activated LoRA: Fine-tuned LLMs for Intrinsics. arXiv:2504.12397.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *CoRR*, abs/2106.09685.

Kishimoto, A.; Bouneffouf, D.; Marinescu, R.; Ram, P.; Rawat, A.; Wistuba, M.; Palmes, P.; and Botea, A. 2022. Bandit limited discrepancy search and application to machine learning pipeline optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36(9), 10228–10237.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. *CoRR*, abs/2104.08691.

Li, L.; Jamieson, K. G.; DeSalvo, G.; Rostamizadeh, A.; and Talwalkar, A. 2016. Efficient Hyperparameter Optimization and Infinitely Many Armed Bandits. *CoRR*, abs/1603.06560.

Liaw, R.; Liang, E.; Nishihara, R.; Moritz, P.; Gonzalez, J. E.; and Stoica, I. 2018. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*.

Thakur, A. 2024. AutoTrain: No-code training for state-of-the-art models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 419–423. Miami, Florida, USA: Association for Computational Linguistics.

Tomar, S.; Marinescu, R.; Tirupathi, S.; Daly, E.; and Dusparic, I. 2025. AT4TS : AutoTune for Time Series Foundation Models. *Transactions on Machine Learning Research - Forthcoming*.

Weidele, D. K. I.; Afzal, S.; Valente, A. N.; Makuch, C.; Cornec, O.; Vu, L.; Subramanian, D.; Geyer, W.; Nair, R.; Vejsbjerg, I.; et al. 2023. AutoDOViz: Human-Centered Automation for Decision Optimization. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, 664–680.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.