AccelOpt: A Self-Improving LLM Agentic System for AI Accelerator Kernel Optimization

Genghan Zhang[1], Shaowei Zhu[2], Anjiang Wei[1], Zhenyu Song[1], Allen Nie[1], Zhen Jia[2], Nandita Vijaykumar[2,3], Yida Wang[2], Kunle Olukotun[1]

[1]Stanford University [2]Amazon Web Services [3]University of Toronto

***Abstract*—We present AccelOpt, a self-improving large language model (LLM) agentic system that autonomously optimizes kernels for emerging AI acclerators, eliminating the need for expert-provided hardware-specific optimization knowledge. AccelOpt explores the kernel optimization space through iterative generation, informed by an optimization memory that curates experiences and insights from previously encountered slow-fast kernel pairs. We build NKIBench, a new benchmark suite of AWS Trainium accelerator kernels with varying complexity extracted from real-world LLM workloads to evaluate the effectiveness of AccelOpt. Our evaluation confirms that AccelOpt's capability improves over iterations, boosting the average percentage of peak throughput from 49% to 61% on Trainium 1 and from 45% to 59% on Trainium 2 for NKIBench kernels. Moreover, AccelOpt is highly cost-effective: using open-source models, it matches the kernel improvements of Claude Sonnet 4 while being $26\times$ cheaper. The code is open-sourced at https://github.com/zhang677/AccelOpt and the full-length paper was published in MLSys 2026 [26].**

I. Problem Statement

Kernel optimization remains challenging even on mature architectures such as GPUs. After NVIDIA released H100 in 2022, attention kernels reached only $\sim$37% of peak performance after one year [8], and required another year to approach 85% [21]. High efficiency depends on a complex interaction between workload properties, memory hierarchies, parallelism, and architecture-specific constraints, making empirical tuning and large search spaces unavoidable [12], [25], [27]. These challenges are amplified for emerging AI accelerators, whose architectures diverge from GPUs and lack established optimization heuristics or performance intuition [9], [11], making kernel optimization a recurring challenge for each new production accelerator [6], [16], [18], [20].

II. Related Work

This work targets kernel optimization for AWS Trainium, a widely deployed accelerator that exemplifies these challenges [2]. Trainium is programmed using the Neuron Kernel Interface (NKI), a Python-embedded kernel language with a relatively new hardware and software stack [3]. As a result, developers lack the mature optimization recipes available on GPUs [22]. This motivates scalable, automated kernel optimization methods as workloads continue to evolve [13].

Prior work shows that LLMs can generate correct, high-performance kernels across GPUs, TPUs, and NPUs [1], [7], [10], [14], [15], [17], [19], [23], [24]. Given the limited Trainium-specific optimization knowledge, our goal is to evaluate whether an LLM-based system can autonomously navigate the optimization space without relying on human-engineered heuristics or preexisting tuning examples.

III. Design

This work makes the following contributions: **(1)** AccelOpt, the first self-improving LLM agentic kernel-optimization system for emerging AI accelerators that combines search with memory accumulation. To the best of our knowledge among open-source systems, it is also the first on emerging accelerators that requires neither expert, hardware-specific optimization knowledge nor predefined optimization recipes. **(2)** NKIBench, the first benchmark suite for NKI kernel optimization on Amazon Trainium, with kernels derived from real-world LLM workloads; it evaluates performance against Trainium's theoretical peak rather than only relative speedups (which depend on baseline choice). **(3)** AccelOpt achieves substantial optimizations on NKIBench; the current 14-kernel suite provides a foundation for future NKI optimization research. We also show open-source LLMs (gpt-oss-120b, Qwen3-Coder-480B-A35B-Instruct-FP8) can deliver comparable gains at much lower cost than Claude Sonnet 4.

IV. Evaluations

a) Cost effectivness: As shown in fig. 2, using open-source LLMs, AccelOpt improves the average percentage of peak throughput from 49% to 61% on Trainium 1, which is on par with Claude Sonnet 4 (thinking mode) but $26\times$ cheaper, and from 45% to 59% on Trainium 2 at a similar cost across all tasks in NKIBench.

b) Ablation study of AccelOpt components: As shown in fig. 3, beam search is more effective than repeated sampling for inference-time scaling; adding optimization memories changes cost efficiency in mixed ways, but does not materially improve the best-found kernels given sufficient sampling compared to beam search alone.

c) Comparison with human experts: To quantify how close AccelOpt comes to expert-level results, we tracked its progress on two kernels with human-optimized reference versions. **(1) Mamba.** The NKI tutorial [5] provides three progressively faster human versions, reaching 28.4%, 30.1%, and 52.7% of peak throughput. Starting from the same baseline

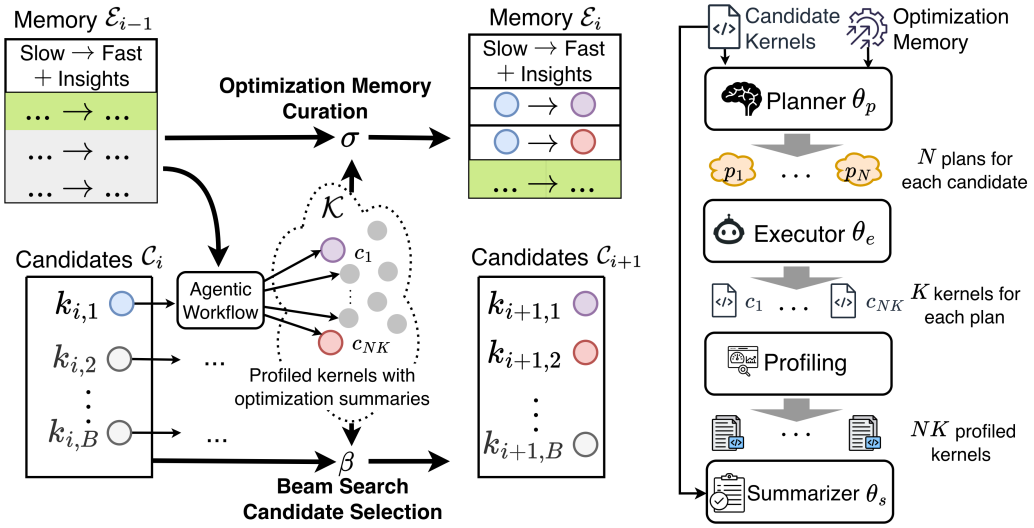

Fig. 1. AccelOpt uses *beam search* to explore the Trainium kernel optimization space through iterative generation of new kernels based on old ones, while retaining top-performing candidate kernels for consideration. When generating new kernels in each iteration, AccelOpt employs a three-component agentic workflow—planner, executor, and summarizer—that mimics how human experts approach the problem. Profiles of the generated kernels are obtained through a distributed profiling service and subsequently used to curate an *optimization memory*. This memory stores a selection of past exploration experiences, including key code changes that resulted in slow-to-fast kernel transformations, along with LLM-generated summaries of general optimization insights. The optimization memory then serves to inspire new optimization strategies in future iterations.

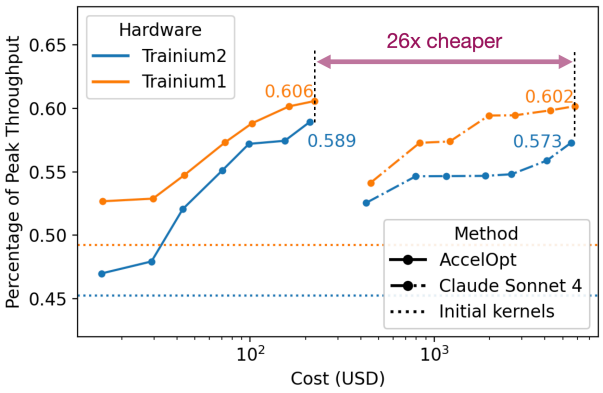

Fig. 2. Compare AccelOpt using open-source LLMs with repeated sampling of Claude Sonnet 4 on Trainium 1 and 2.

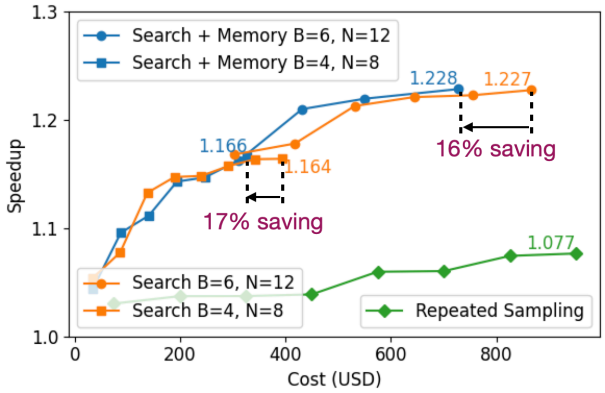

Fig. 3. Geometric mean of best speedup achieved up to a certain iteration across all tasks obtained through repeated sampling, beam search, and beam search + optimization memory. B is the number of candidates, and N is the number of plans for each candidate.

(28.4% of peak), AccelOpt autonomously improved the kernel to 54.6% of peak, which is 1.04× the best expert result (52.7%). Moreover, the generated kernel used a different loop order than the best human. **(2) RoPE.** The initial RoPE kernel was adopted from nki-samples [4], which provides one version of RoPE (21.1% of peak). Starting from this version, AccelOpt improved performance to 29.6% of peak, a 1.4× speedup over the human reference.

d) Educational Impact: In Stanford CS149 Fall 2025, a graduate-level parallel computing course, we used AccelOpt to optimize a Conv2D kernel outside of NKIBench and achieved 48.8% of peak throughput starting from last year's reference implementation (9.54%). Based on the optimization proposed by AccelOpt, we designed an extra credit problem. 33.6% of 131 teams of students successfully conquered the challenge. Throughout the process, the students learned two principles: (1) transforming sequential temporal iteration to parallel spatial execution, and (2) specialization for workload under hardware constraints. This example underscores the generality of AccelOpt beyond NKIBench and highlights the educational impact of LLM-assisted kernel optimization.

Together, these results demonstrate that AccelOpt can exceed expert-level performance. This stems from AccelOpt's scalability: human experts optimize a handful of kernels sequentially, while AccelOpt can explore many in parallel.

V. Conclusion

This paper presents AccelOpt, the first self-improving LLM agentic system for kernel optimization on emerging AI accelerators that combines search with memory accumulation. We demonstrate that combining inference-time scaling with optimization memory enables LLM agents to autonomously optimize real-world Trainium kernels in NKIBench, our curated benchmark suite, without requiring expert optimization knowledge. Through systematic ablation studies, we confirm the effectiveness of beam search and optimization memory in obtaining high-performing kernels with improved cost efficiency. We also find that open-source models achieve higher cost efficiency than leading proprietary coding models for this task. Overall, AccelOpt and NKIBench provide a promising foundation for automated kernel optimization on emerging AI accelerators.

References

[1] L. A. Agrawal, S. Tan, D. Soylu, N. Ziems, R. Khare, K. Opsahl-Ong, A. Singhvi, H. Shandilya, M. J. Ryan, M. Jiang *et al.*, "Gepa: Reflective prompt evolution can outperform reinforcement learning," *arXiv preprint arXiv:2507.19457*, 2025.

[2] AWS, "AWS Trainium: AI Training Accelerator," https://aws.amazon.com/ai/machine-learning/trainium/, 2025, accessed: 2025-10-25.

[3] ——. (2025) Neuron kernel interface (nki) (beta) 2.20. [Online]. Available: https://awsdocs-neuron.readthedocs-hosted.com/en/latest/nki/nki_rn.html#neuron-kernel-interface-nki-beta-2-20

[4] ——. (2025) Nki samples. [Online]. Available: https://github.com/aws-neuron/nki-samples/blob/main/src/nki_samples/tutorials/rotary/rotary_nki_kernels.py

[5] ——. (2025) Nki tutorials. [Online]. Available: https://github.com/aws-neuron/nki-samples/blob/main/src/nki_samples/tutorials/fused_mamba/mamba_nki_kernels.py

[6] M. Azure. (2024) Maia. [Online]. Available: https://azure.microsoft.com/en-us/blog/azure-maia-for-the-era-of-ai-from-silicon-to-software-to-systems/

[7] C. Baronio, P. Marsella, B. Pan, S. Guo, and S. Alberti, "Kevin: Multi-turn rl for generating cuda kernels," *arXiv preprint arXiv:2507.11948*, 2025.

[8] T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," *arXiv preprint arXiv:2307.08691*, 2023.

[9] S. Fang, H. Chen, N. Zhang, J. Li, H. Meng, A. Liu, and Z. Zhang, "Dato: A task-based programming model for dataflow accelerators," *arXiv preprint arXiv:2509.06794*, 2025.

[10] C. Hong, S. Bhatia, A. Cheung, and Y. S. Shao, "Autocomp: Llm-driven code optimization for tensor accelerators," *arXiv preprint arXiv:2505.18574*, 2025.

[11] O. Hsu, A. Rucker, T. Zhao, V. Desai, K. Olukotun, and F. Kjolstad, "Stardust: Compiling sparse tensor algebra to a reconfigurable dataflow architecture," in *Proceedings of the 23rd ACM/IEEE International Symposium on Code Generation and Optimization*, 2025, pp. 628–643.

[12] Z. Jia, O. Padon, J. Thomas, T. Warszawski, M. Zaharia, and A. Aiken, "Taso: optimizing deep learning computation with automatic generation of graph substitutions," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019, pp. 47–62.

[13] S. Kim, C. Hooper, T. Wattanawong, M. Kang, R. Yan, H. Genc, G. Dinh, Q. Huang, K. Keutzer, M. W. Mahoney *et al.*, "Full stack optimization of transformer inference: a survey," *arXiv preprint arXiv:2302.14017*, 2023.

[14] R. T. Lange, Q. Sun, A. Prasad, M. Faldor, Y. Tang, and D. Ha, "Towards robust agentic cuda kernel benchmarking, verification, and optimization," *arXiv preprint arXiv:2509.14279*, 2025.

[15] X. Li, X. Sun, A. Wang, J. Li, and C. Shum, "Cuda-l1: Improving cuda optimization via contrastive reinforcement learning," *arXiv preprint arXiv:2507.14111*, 2025.

[16] Meta. (2024) Mtia. [Online]. Available: https://ai.meta.com/blog/next-generation-meta-training-inference-accelerator-AI-MTIA/

[17] A. Novikov, N. Vũ, M. Eisenberger, E. Dupont, P.-S. Huang, A. Z. Wagner, S. Shirobokov, B. Kozlovskii, F. J. Ruiz, A. Mehrabian *et al.*, "Alphaevolve: A coding agent for scientific and algorithmic discovery," *arXiv preprint arXiv:2506.13131*, 2025.

[18] OpenAI, "Openai-designed ai accelerators," 2025. [Online]. Available: https://openai.com/index/openai-and-broadcom-announce-strategic-collaboration/

[19] A. Ouyang, S. Guo, S. Arora, A. L. Zhang, W. Hu, C. Ré, and A. Mirhoseini, "Kernelbench: Can llms write efficient gpu kernels?" *arXiv preprint arXiv:2502.10517*, 2025.

[20] Qualcomm. (2025) Ai200 and ai250. [Online]. Available: https://www.qualcomm.com/news/releases/2025/10/qualcomm-unveils-ai200-and-ai250-redefining-rack-scale-data-cent

[21] J. Shah, G. Bikshandi, Y. Zhang, V. Thakkar, P. Ramani, and T. Dao, "Flashattention-3: Fast and accurate attention with asynchrony and low-precision," *Advances in Neural Information Processing Systems*, vol. 37, pp. 68 658–68 685, 2024.

[22] V. Thakkar, P. Ramani, C. Cecka, A. Shivam, H. Lu, E. Yan, J. Kosaian, M. Hoemmen, H. Wu, A. Kerr, M. Nicely, D. Merrill, D. Blasig, F. Qiao, P. Majcher, P. Springer, M. Hohnerbach, J. Wang, and M. Gupta, "CUTLASS," Jan. 2023. [Online]. Available: https://github.com/NVIDIA/cutlass

[23] A. Wei, A. Nie, T. S. F. X. Teixeira, R. Yadav, W. Lee, K. Wang, and A. Aiken, "Improving parallel program performance with LLM optimizers via agent-system interfaces," in *Forty-second International Conference on Machine Learning*, 2025. [Online]. Available: https://openreview.net/forum?id=3h80HyStMH

[24] J. Woo, S. Zhu, A. Nie, Z. Jia, Y. Wang, and Y. Park, "Tritonrl: Training llms to think and code triton without cheating," *arXiv preprint arXiv:2510.17891*, 2025.

[25] M. Wu, X. Cheng, S. Liu, C. Shi, J. Ji, M. K. Ao, P. Velliengiri, X. Miao, O. Padon, and Z. Jia, "Mirage: A {Multi-Level} superoptimizer for tensor programs," in *19th USENIX Symposium on Operating Systems Design and Implementation (OSDI 25)*, 2025, pp. 21–38.

[26] G. Zhang, S. Zhu, A. Wei, Z. Song, A. Nie, Z. Jia, N. Vijaykumar, Y. Wang, and K. Olukotun, "Accelopt: A self-improving llm agentic system for ai accelerator kernel optimization," *arXiv preprint arXiv:2511.15915*, 2025.

[27] L. Zheng, C. Jia, M. Sun, Z. Wu, C. H. Yu, A. Haj-Ali, Y. Wang, J. Yang, D. Zhuo, K. Sen *et al.*, "Ansor: Generating {High-Performance} tensor programs for deep learning," in *14th USENIX symposium on operating systems design and implementation (OSDI 20)*, 2020, pp. 863–879.