The Cost of AI-Assisted Coding: Energy vs. Accuracy in Language Models*

Negar Alizadeh[1], Boris Belchev[2], Nishant Saurabh[1] and Patricia Kelbert[3]

[1]Utrecht University, Utrecht, The Netherlands
[2]University of Twente, Enschede, The Netherlands
[3]Fraunhofer IESE, Kaiserslautern, Germany

1. Introduction and Motivation

Generative Large Language Models (LLMs) have become widely accessible since the release of ChatGPT in late 2022 [1], and their adoption nearly doubled in under six months [2]. In addition, the majority of developers find code-specific AI models beneficial and have integrated them into their daily workflows. [3, 4].

Even though these AI tools are accessible through third-party APIs, client companies are mainly concerned about data privacy, security, and subscription costs. This motivates the use of locally deployed open-access language models. One approach for deploying LLMs locally is to utilize a flagship GPU with sufficient memory optimized for Deep Learning (DL) applications and to set up an open-access LLM on it. However, since these GPUs are not affordable for everyone, an alternative solution could be using compressed and quantized models that can run on smaller GPUs or even large CPUs.

At the same time, the energy cost of LLMs, especially during inference, has become a growing concern due to financial and environmental impacts [5, 6]. Furthermore, recent studies evaluating code-centric LLMs have mainly focused on performance in terms of accuracy, often overlooking their energy footprint.[7, 8].

The goal of this study is to investigate the energy consumption of using LLMs during the inference phase in typical software development tasks, namely code generation, bug fixing, docstring generation, and test case generation. We selected these 4 tasks because they are widely used by developers [9] and frequently addressed in software engineering research involving deep learning [10]. They also represent the software development lifecycle, from implementation and documentation to testing and maintenance.

In particular, we address the following research questions:

RQ1 How does energy usage vary across software-related tasks?

RQ2 Is there a trade-off between energy efficiency and accuracy?

RQ3 Which model characteristics influence energy consumption?

RQ4 What are the performance differences between general-purpose models and code-specific models?

2. Methodology and Results

Our experimental design involves evaluating 18 language model families across three precision formats on two real-world hardware setups: a high-end A100 GPU and a consumer-grade RTX3070 GPU. Model

*Accepted at the 22nd International Conference on Mining Software Repositories (MSR 2025)
n.s.alizadeh@uu.nl (N. Alizadeh); b.belchev@student.utwente.nl (B. Belchev); n.saurabh@uu.nl (N. Saurabh); patricia.kelbert@iese.fraunhofer.de (P. Kelbert)

selection was based on their popularity on Hugging Face, availability, and the reputability of their creators. Running an LLM locally requires a compatible runtime to handle inference across devices. We chose Ollama based on its popularity on GitHub and ease of switching between models. All evaluations are conducted on the HumanEvalPack dataset [11], with top-p set to 0.95 and temperature to 0.1, following prior studies [12, 13, 14]. As for the evaluation metric, we report test coverage and correctness for test generation and pass@1 for the rest.

Our findings indicate that a model's energy consumption is directly affected by the software development task it performs, with notable performance variations between tasks. Therefore, selecting models based on the tasks they are expected to handle is key to reducing the energy footprint. In contrast, energy usage per generated token remains consistent across tasks for a model. Additionally, the total energy consumed by each model is strongly correlated with its architectural characteristics, suggesting that some aspects of an LLM's architecture can help estimate its efficiency, given that the average size of the outputs can be anticipated. Finally, we observed that energy consumption and accuracy do not always require a compromise, as larger models often have a significantly higher energy footprint while performing similarly, or even being outperformed by smaller models in terms of accuracy. Finally, "Coding"-specific models can be more accurately described as "code generation" models. Our results suggest that fine-tuning models for other tasks, such as docstring generation and bug fixing, is a potential research avenue.

To improve generalizability, future work could include more programming languages and use benchmarks with complex, real-world tasks rather than simple Python functions. Currently, we are exploring ways to predict model efficiency based on architectural features, to reduce the need for costly experiments.

References

[1] C. Ebert, P. Louridas, Generative AI for software practitioners, IEEE Softw. 40 (2023) 30–38. doi:10.1109/MS.2023.3265877.

[2] Microsoft Corporation, AI at Work Is Here—Now Comes the Hard Part, "https://www.microsoft.com/en-us/worklab/work-trend-index/ai-at-work-is-here-now-comes-the-hard-part", 2024. Accessed: 2024-11-05.

[3] Stack Overflow, Developers get by with a little help from ai: Stack overflow knows code - assistant pulse survey results, https://stackoverflow.blog/2024/05/29/developers-get-by-with-a-little-help-from-ai-stack-overflow-knows-code-assistant-pulse-survey-results/, 2024. Accessed: 2024-11-08.

[4] GitHub, Research: Quantifying github copilot's impact in the enterprise with accenture, https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-in-the-enterprise-with-accenture/, 2024. Accessed: 2024-11-08.

[5] E. Strubell, A. Ganesh, A. McCallum, Energy and policy considerations for deep learning in NLP, CoRR abs/1906.02243 (2019).

[6] A. Lacoste, A. Luccioni, V. Schmidt, T. Dandres, Quantifying the carbon emissions of machine learning, CoRR abs/1910.09700 (2019).

[7] Z. Zheng, K. Ning, Y. Wang, J. Zhang, D. Zheng, M. Ye, J. Chen, A survey of large language models for code: Evolution, benchmarking, and future trends, CoRR abs/2311.10372 (2023).

[8] J. Jiang, F. Wang, J. Shen, S. Kim, S. Kim, A survey on large language models for code generation, CoRR abs/2406.00515 (2024).

[9] M. Khemka, B. Houck, Toward effective AI support for developers: A survey of desires and concerns, Commun. ACM 67 (2024) 42–49. doi:10.1145/3690928.

[10] C. Watson, N. Cooper, D. Nader-Palacio, K. Moran, D. Poshyvanyk, A systematic literature review on the use of deep learning in software engineering research, ACM Trans. Softw. Eng. Methodol. 31 (2022) 32:1–32:58. doi:10.1145/3485275.

[11] N. Muennighoff, Q. Liu, A. Zebaze, Q. Zheng, B. Hui, T. Y. Zhuo, S. Singh, X. Tang, L. von Werra,

S. Longpre, Octopack: Instruction tuning code large language models, CoRR abs/2308.07124 (2023).

[12] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, et al., Starcoder: may the source be with you!, Trans. Mach. Learn. Res. 2023 (2023).

[13] A. Lozhkov, R. Li, L. B. Allal, F. Cassano, J. Lamy-Poirier, N. Tazi, A. Tang, D. Pykhtar, J. Liu, Y. Wei, et al., Starcoder 2 and the stack v2: The next generation, CoRR abs/2402.19173 (2024).

[14] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. Bhatt, C. Canton-Ferrer, A. Grattafiori, W. Xiong, A. Défossez, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, G. Synnaeve, Code llama: Open foundation models for code, CoRR abs/2308.12950 (2023).