The Unaccounted Carbon Cost of AI-Assisted Software Engineering: A Hidden Debt and Sustainability Challenge

Nelly Nicaise Nyeck Mbialeu, Benjamin Leiding
Institute for Software and Systems Engineering
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
e-mail: {nelly.nicaise.nyeck.mbialeu | benjamin.leiding}@tu-clausthal.de

***Abstract*—Software engineering is experiencing the impact of AI on its productivity through rapid code generation, code fixes, and workflow automation. However, there is a hidden cost to this convenience, namely a growing double debt of carbon emissions and technical inefficiencies that jeopardise sustainability. Carbon debt is discussed in this paper, referring to the invisible and cumulative environmental damage resulting from the frequent use of AI-driven tools. AI sustainability discussions often overlook the impact of inference phase emissions in this field, where productivity tools lack built-in insights that measure hidden, accumulated environmental burdens. There is a lack of a conceptual and structured method to incentivise carbon debt. This paper conceptually illustrates the negative contribution of AI-assisted development tools, leading to pragmatic mitigation strategies and a preliminary formalization of a measurable sustainability framework for AI-driven development workflows.**

Keywords-carbon debt; sustainability; technical debt; AI DevTools; green software engineering

I. Introduction

The swift adoption of Artificial Intelligence (AI) into software engineering has fundamentally changed how software is designed, developed, tested, and maintained. Tools such as GitHub Copilot, automated bug fixers, and intelligent testing frameworks now assist developers at nearly every stage of the software lifecycle, ensuring productivity through generating code, debugging, and streamlining workflows. However, their downside is an invisible cost due to the constant use of computing power to support large-scale AI models running in the background. When software engineers use AI-generated suggestions or machine learning tests, the associated processes are executed in high-energy-consuming data centres, which contribute to a significant environmental footprint. In areas where these centres use fossil fuels or operate with passive regulatory oversight, as revealed by Elon Musk's xAI data centre, which used unauthorized gas turbines [1], the carbon intensity of AI services continues to increase, exacerbating the hidden carbon debt. The infrastructure that powers AI-driven development tools is resource-intensive, and its environmental impact extends far beyond electricity consumption. Graphic Processing Units (GPUs), the computational core of these AI tools, are produced using rare earth minerals and resource-intensive manufacturing techniques with complex supply chains, which are frequently linked to labour issues and environmentally damaging mining activities [2], [3]. Once deployed, these GPUs work in data centres that need massive cooling systems whose operations require a lot of electricity and water to keep running at optimal efficiency [4]. Likewise, the short lifespans of AI devices add to the increasing amount of electronic waste, which worsen climate change and emit harmful substances if not adequately controlled [5]. These lifecycle impact, ranging from extraction to disposal, add up to what can be referred to as a form of carbon debt (a concealed but growing environmental cost associated with the creation and application of AI technologies). While the carbon cost of training huge AI models has received a lot of attention, the accumulated impact of using them during day-to-day software development is less known and rarely acknowledged.

Like financial debt, carbon debt is a hidden expense that eventually needs to be paid back to mitigate environmental impact, as discussed in [6], for the necessity of meeting climate targets. Carbon debt is an imperceptible environmental cost that accumulates over time as a result of adopting energy-intensive AI-driven software engineering methodologies [7]. Similar to the well-known concept of technical debt in software development [8], carbon debt builds subtly and is often overlooked in the short term. Indeed, these debts are interconnected as the same AI tools that accelerate development today also contribute to escalating sustainability risks, which can have significant long-term repercussions if left unaddressed [8]. As such, carbon debt highlights the trade-offs between immediate efficiency gains and future environmental liabilities. Although end users may not be aware of the carbon impact of AI technologies, acknowledging their influence is becoming increasingly essential as these tools are integrated into development workflows. This study provides a conceptual analysis for understanding and mitigating carbon debt in software engineering by proposing a shift in mindset from prioritizing efficiency to embracing environmental responsibility. Additionally, the goal is to educate with ideas or actionable solutions needed to make carbon-aware software development a core part of responsible AI practice and frame the urgency of AI-assisted Software engineering emissions as debt before regulators or climate consequences force our hand.

This paper seeks to answer the following research questions:
RQ1: *How does AI-assisted software development contribute to carbon debt, a hidden form of environmental damage?*
RQ2: *What strategies can be applied to mitigate carbon debt without reinforcing the unethical adoption of AI in SE workflows?*

In software engineering, AI Development Tools (DevTools) are frequently evaluated in terms of productivity and code quality, with their ecological impacts totally ignored and unmeasured. This study introduces carbon debt as a conceptual lens to understand better the long-term consequences and advocates for incorporating environmental criteria, such as the sustainability impact factor, into our assessment of the utility and responsibility of such tools. The value of this contribution is to spark discussion through the conceptual lens of carbon debt and guide future research to explore how this framing might be quantified, modeled, or embedded into development environments.

The paper is structured as follows: Section II illustrates the emissions from AI software engineering tools. Section III portrays the double debt trap and the invisible accumulation of carbon debt. Next, Section IV presents mitigation strategies across the perspectives of multiple stakeholders. The early contours of a sustainability impact assessment are proposed in Section V for future evaluation of the responsibilities of these AI tools. Critical reflections and limitations are discussed in Section VI. Finally, Section VII concludes this study and provides an outlook for further research.

II. Carbon Debt in AI-driven Software Engineering

The increasing integration of AI tools into software development workflows comes with an unclear sustainability concern that is not immediately visible but has long-term consequences for the planet, as it is underexplored. In contrast to financial costs that are obvious in cloud service bills, the AI coding tools increasingly being adopted in development build up carbon emissions [9] shrouded in opacity as invisible carbon debt. GitHub Copilot, Amazon CodeWhisperer, and ChatGPT are examples [10] of these tools that, with each query, assist developers with real-time code suggestions and blocks of logic generation, thus requiring computational resources [11]. Each interaction implies running inference on massive Large Language Models (LLMs) in energy-intensive data centres, resulting in non-trivial energy consumption [12] that scales rapidly with frequent usage. This leads to an unacknowledged environmental burden referred to as carbon debt [13].

A. The Hidden Emissions of Everyday AI-assisted Tools

These tools constantly feed user input to LLMs hosted in cloud infrastructures that consume much energy [12], [14], [15]. The training stage of large models, which is, in fact, energy-intensive, has been the focus of most research on the environmental impact of AI [13], [15]. But, inference (the real-time application of these models each time a developer inputs a line of code or requests code, then gets a suggestion) is instead the primary cause of carbon debt rather than training, as observed by studies quantifying carbon emissions from AI inference [16]–[18]. This continual inference workload [16], is multiplied across thousands of users, Integrated Development Environments (IDEs), and ongoing delivery pipelines, resulting in significant energy usage [19] that remains largely overlooked. As such, in the realm of AI-assisted software engineering, the impact of daily tool usage in terms of long-term carbon emissions is still quite open for investigation. The following are examples of energy-intensive cloud infrastructure that are ingrained in Software Engineering workflows and could be invoked repeatedly during coding:

- Github Copilot
  Every code suggestion GitHub Copilot generates requires inference from a LLM hosted on Microsoft Azure servers because it is frequently used as an AI coding assistance. The model powering Copilot (Codex) is fine-tuned from GPT-3 and probably used less energy during training; nonetheless, the estimated emissions of GPT-3 of about 500 tonnes of CO_2 [20] serve as a valuable benchmark for measuring the environmental impact of LLM-based tools. While Microsoft does not disclose precise figures, each suggestion is reported by community estimates to consume about 0.002kWh of energy, equivalent to roughly 1.2g CO_2 per inference [21]. This is in line with broader estimates of small-scale AI inference tasks energy use [13], [15], [17], [18]. Even though this reported value seems low in isolation, the total carbon emissions from ongoing, real-time inferences made during daily software engineering tasks could add up to a non-negligible environmental cost. This observation is consistent with broader research in the field, which shows how, when scaled to millions of operations, seemingly minor per-inference energy costs can substantially impact the overall carbon footprint [13].
- AI Testing Tools
  These tools are essential to CI/CD pipelines as they enhance developer productivity. Several automated test generators, such as EvoSuite [22] and DiffBlue [23], often generate redundant or inefficient test cases. The authors in [24] show that automated test generation using EvoSuite can produce up to 28% low cohesion and approximately 50% high coupling test methods even after test minimization. This observation suggests that many generated test cases are functionally redundant, which implies more execution and more energy consumption, as they do not improve code reliability but still burn energy. So, when such tools run frequently, the compounded energy from computing power used for testing quickly becomes significant [10], [12]. Additionally, prior studies have shown that continuous integration systems can amplify energy use by an order of magnitude (potentially 10x) when augmented with tools like test generation and fault localisation [25], [26]. The energy footprint of CI/CD workflows could rise rapidly if AI tools result in more test inefficiencies. Like the emission from Copilot, this waste is invisible to developers and silently adds to the carbon debt.

In contrast to broadly applicable AI technologies like ChatGPT, the aforementioned AI tools alleviate carbon debt more through regular integration within IDEs and pipelines as a result of their widespread adoption. These examples above highlight a critical blind spot: unlike performance measurements (such as latency and accuracy), which are clearly visible, developers do not have feedback mechanisms to identify

the carbon cost of using AI tools in daily software engineering operations. Reliable carbon accounting for AI tools remains scarce, highlighting the need for transparent and standardized emission metrics for both training and inference phases.

B. Why Carbon Debt is a Debt

Debt is the commitment to pay back funds or resources, often with added interest, as defined by the Oxford English Dictionary [27]. Beyond finance, the term is often used in a metaphorical sense to refer to hidden accumulated costs that require future repayment, like environmental or technical debt. As such, the carbon debt of AI-assisted software engineering works similarly. Global threats are increasing exponentially with higher temperatures, according to the Intergovernmental Panel on Climate Change (IPCC) [28], which also notes that economic impacts from climate change are being tracked in energy, agriculture, and other vulnerable sectors [28]. As such, carbon emissions are increased by carbon debt from widespread computational operations, such as the real-time use of AI tools that make high energy consumption, which eventually contributes to the global emission budget, exacerbating the threats. The IPCC (2023) also emphasizes that postponing mitigation efforts is expected to increase future costs, including infrastructure damage and health-related impacts. Highlighting the importance of early intervention in mitigating carbon debt in AI-assisted software engineering. Similar to financial debt bearing compound interest, the seemingly negligible energy cost of each inference adds up to cumulative emissions through the numerous daily operations, which causes the carbon debt of AI-assisted tools to grow exponentially [9], [10]. For illustrative purposes, if 30% of the approximately 26 million developers [29] adopted AI coding tools at 50 suggestions per day, and assuming an estimated 0.002 kWh per suggestion [21]. The annual energy use could reach approximately 285 million kWh, which could lead to annual carbon emissions exceeding 100000 tonnes [30], that is comparable roughly to the emissions produced by 20000-25000 passenger vehicles each year [31]. These illustrative projections display the cumulative climate cost of real-time inference and are highly sensitive to adoption rates and infrastructure efficiency. The tech sector frequently presents AI as environmentally friendly (i.e., intrinsically "green") because of data centres that are powered by renewable energy. But this information is quite misleading. Firstly, cloud providers like Microsoft that are committed to achieving 100% renewable energy by 2025 through annual purchases and matching over 95% of its Scope 2 emissions using renewable energy instruments like Power Purchase Agreement (PPA) and Renewable Energy Certificate (REC) [32]. But Microsoft's 2022 report shows that, on an hourly basis, just 60% of its electricity use came from carbon-free sources, underscoring the difference between energy accounting and actual clean energy (real carbon-free) usage [32]. Secondly, training a large LLM, like GPT-3, has a carbon impact of about 552 metric tonnes of CO_2; nonetheless, this number does not include emissions from the construction of data centres or the manufacturing of GPUs [20]. This upfront carbon debt, which is paid before any inference is ever made, added to the overlooked embodied emissions, contributes highly to the overall environmental impact as increased productivity from faster code generation leads to higher energy use.

III. Technical Debt Analogy and Pipeline

Technical debt occurs when developers use shortcuts or suboptimal solutions to achieve short-term goals, which eventually increases complexity and maintenance expenses in the long run [8]. This section observes the role of AI in software engineering in relation to technical debt within the context of sustainability. It examines how the lifecycle of technical debt is well related to carbon debt.

A. AI's Hidden Tax on Code Quality

Developers identify inefficiencies in AI-generated code, as revealed in a study [33], where approximately 40% of code snippets suggested by Copilot contain security vulnerabilities. These flaws increase future maintenance workloads and reduce code quality, which may require a lot of energy use for the necessary later fixes. According to studies, code generated by AI-assisted tools often contains structural issues like poor modularity and tight coupling ("code smells") [33], [34], which can make it challenging to maintain and result in future updates that require high energy demands. These problems are similar to those of traditional technical debt but with a carbon twist.

B. Maintenance Burden of Hidden Cost

Due to challenges in code maintainability and security vulnerabilities brought by AI-suggested code, post-deployment fixes for AI-assisted software projects have been found to occur frequently [33], [35]. This goes to show the frequent need for fixes and remediation, thereby resulting in escalating energy demands indirectly linked to more carbon debt. Thus, AI-assisted development works on a carbon credit basis, whereby rising emissions from subsequent maintenance balance out the energy savings of quick initial coding. The industry lacks tools to account for this delayed sustainability responsibility, making the environmental benefits of AI's productivity questionable.

Both categories portray the double debt trap, in which AI-assisted tools create technical debt that silently inflates carbon debt. Whereby code quality tradeoffs lead to higher maintenance requirements that contribute to a loop of increasing energy consumption and carbon impact. In contrast, other technical debt dimensions, such as security and scalability debt, are typically resolved with localised solutions (such as patching a single library) instead of systemic energy loss. In essence, just as technical debt prioritises speed over quality, carbon debt similarly compromises sustainability for productivity.

The analysis of emissions from AI-SE tools (Section II) and their cumulative consequences through technical debt (Section III) yields insights into **RQ1**, illustrating that carbon debt builds up undetected throughout the lifecycle of AI-assisted software development.

IV. Strategies to Mitigate Carbon Debt

The first step in addressing the issue of carbon debt, which is a hidden cost of AI-assisted software engineering, is to identify targeted strategies and practical solutions inspired by the technical debt analogy. To avoid relegating sustainability to a secondary concern, proactive approaches are needed to ensure that AI development is in line with long-term environmental goals. Rather than providing concrete technical solutions, these approaches serve as foundational concepts for software design, educational initiatives, and future research endeavors.

A. Operationalizing Carbon Awareness

One key challenge to lowering carbon debt in software development is its invisibility. Usually, developers don't get feedback on the energy costs or carbon emissions associated with using AI tools. So, to improve awareness, carbon transparency features could be incorporated, relying on prior research on machine learning emissions tracking [18], [36]–[38].

- *Real-time emissions dashboard* to display approximate carbon emissions, for example, IDE plugins such as CodeCarbon [38], per AI suggestions, or after code completions per day.
- *Eco-modes* that limit the use of AI tools or give priority to suggestions that are energy-effective.
- *Contextual pop-ups* that alert developers when behaviours like repeated Copilot requests exceed sustainability thresholds.

These will help developers strike a balance between productivity and sustainability by highlighting the environmental impact of AI and recognizing carbon impact as an essential element of software quality.

B. Context-Aware AI Tool Usage

Selective invocation is a significant mitigating technique since the usage of AI tools varies depending on their impact. This means avoiding unnecessary applications like boilerplate code and deploying large models for complex tasks to limit useless inference overhead [12], [13]. Some sustainability-focused practices:

- Prompt engineering, which will reduce energy use by, for example, generating a low memory algorithm [34], [39].
- AI-assisted refactoring of carbon-heavy patterns, such as nested loops, to improve efficiency, and also flagging excessive energy use by setting CO_2 limits daily, for example.

Giving developers usage reports promotes introspection and effective adoption of tools.

C. Integrating Sustainability in Education

A cultural shift is necessary to mitigate carbon debt, as developers cannot efficiently handle what remains beyond awareness. Incorporating sustainability principles in software engineering education as AI becomes more ingrained in development practices [40], [41].

- Workshops on Green AI and techniques to audit AI tools for carbon efficiency.
- Hands-on exercises on energy consumption of both manual and AI-assisted tasks.
- Reflective workshops on environmental compromises in software design.

D. Policy Levers

Carbon accountability could be enforced, such as expanding the EU's Carbon Border Adjustment Mechanism to include cloud-based AI capabilities.

E. Advocating for Purpose-Limited Technology

Energy-intensive AI tools that offer only marginal benefits could be rejected or stop being used when their core justification remains weak. Thereby promoting simple, sustainable solutions like writing code manually to create maintainable software rather than using possibly redundant recommendations. Sometimes, the most moral and environmentally responsible course of action for a developer is to choose not to adopt technological advancement when the ecological costs are unjustifiable.

Without visibility into emissions, AI DevTools leave careful users in the dark (unaware of their ecological impact). Therefore, reforming the system is necessary for significant change, while individual efforts to reduce carbon debt are essential.

V. Towards Sustainability Impact Assessment

With the aim of understanding the environmental costs of AI-supported software engineering, this study presents the concept of carbon debt, which lays the groundwork for developing visible, measurable, and actionable assessment mechanisms. This concept can be considered as a focused instance of the broader Sustainability Impact Factor (SIF) framework suggested by Lawrenz et al. [42]. The authors propose measuring the fixed and variable environmental consequences of digital tools and services to promote the implementation of service-level sustainability reporting in circular ecosystems. It is worth acknowledging the significant cost associated with training large models and manufacturing the supporting hardware (i.e., *fixed sustainability impact*). However, in the present context, the focus is on the *variable sustainability impact* resulting from the ongoing, cumulative emissions from the regular use of AI tools, which accumulate invisibly over the daily interactions of millions of developers. As the systematic tracking of these emissions could form part of a measurable component within a tool-specific SIF that can guide responsible usage and development behaviours. The following Tab I suggests preliminary criteria intended for initial discussion regarding the SIF for AI DevTools:

The mitigation strategies proposed in Section IV collectively address **RQ2** and respond to the double debt issue of aiming to maintain productivity while minimizing both environmental and technical debt associated with AI tools. Section V extends the discussion toward future impact tracking models.

VI. Discussion and Limitations

Generally, in software engineering, emphasis is laid on fairness, bias, transparency, and privacy, whilst environmental

TABLE I. PRELIMINARY CONCEPTUAL DIMENSIONS PROPOSAL OF A SUSTAINABILITY IMPACT FACTOR FRAMED FROM THE CARBON DEBT CONCEPT.

Dimension	Illustrative Metric	Rationale
Operational Efficiency	CO_2e per 1000 completions or test executions	Measures the continuous cost of routine tool usage
Model Training Efficiency	Total carbon emissions during pretraining and fine-tuning	symbolizes the environmental impact of AI model development
Usage Intensity	Average daily invocations per developer	Shows the level of integration and influence the tool has in workflows
Energy Source	Ratio of renewable to fossil-powered inference	Distinguishes between greener and carbon-intensive AI operations
Hardware Lifecycle	Average GPU replacement cycle alongside e-waste per model	Highlight the physical resource impacts and disposal challenges
Transparency Practices	Availability of data on model size, energy use, and emissions	Encourages accountability in sustainability reporting

impact is often ignored [43]. To guarantee a really responsible approach to AI, software development must address carbon awareness and sustainability as equally essential components, as they indicate our broader interaction with digital infrastructure and planetary boundaries. Instead of being an afterthought, carbon impact must be a visible, measurable indicator. As AI DevTools continue to be integrated into daily workflows (e.g., Copilot, CodeWhisperer, testing suites), their substantial energy impact over time [13], [20] must be a shared responsibility between developers, hosting companies, regulators, and educational institutions. The actors mentioned above should broaden their view of accountability related to the impact of AI in software engineering on society and the environment. In an era of climate urgency, passive observation is unacceptable, as it is essential to ensure that advancement does not come at the price of sustainability.

A more effective approach worth considering is to avoid utilising AI when the assumption of its benefits could be challenged. Techno-critical scholars like Schmachtenberger [44] argue that adopting technological advancements must start with a convincing, fact-based argument that proves that their advantages outweigh their harm while justifying that the harms are reasonable. From this perspective, harm mitigation is not enough compared to not adopting these energy-intensive AI tools, as non-adoption is the most ethical and carbon-conscious decision to begin with.

Throughout this study, increasing awareness about Carbon Debt in AI-assisted Software Engineering is the goal. But it is also important to recognise a number of limitations present here:

- Although studies show that AI technologies produce emissions when inferring, this research does not provide empirical energy values nor accurate measurements of carbon implications in development processes.
- The mitigation strategies proposed were all based on the assumption that AI will remain embedded in software workflows (currently reflecting industry trends). They can make a difference, but would not solve the carbon debt problem, and as a result, a "non-use" AI solution was included but not fully explored in its radicality.
- This study acknowledges technical aspects of carbon debt but does not explore issues such as organisational, economic, and political-economic considerations that also influence AI adoption and provider infrastructure.

VII. CONCLUSION AND FUTURE WORK

The increasing integration of AI-assisted software engineering tools has unveiled an invisibly unacknowledged environmental cost, namely carbon debt. Every output produced by AI is dependent on energy-intensive infrastructure, which results in cumulative emissions that are invisible but increase with usage. This debt, left unmanaged, would accumulate over time and threaten the viability of our ecosystem, even though it does not damage the code. This study defines the concept of carbon debt, inspired by technical debt as a prism through which environmental costs of AI can be viewed. Several strategies to identify and reduce carbon debt were mentioned, such as making the invisible visible, selective AI tool usage, prioritizing long-term sustainability, distributing responsibility, and providing developer education with an emphasis on sustainability. Bringing forth the fact that carbon awareness should be part of responsible AI components in software design. Notwithstanding, the strategy of technological minimalism is mentioned. A cultural and structural shift is necessary for the future. We need toolmakers to stop considering it as an afterthought, developers to follow responsible and reflective practices, educators to equip the next generation with, for example, green coding, and policymakers to create policies that make sustainability the profitable choice. Moreover, a preliminary proposal was discussed on how carbon debt could evolve into a traceable and measurable system (conceptually inspired by previous research in circular economy modelling), thereby promoting accountability in AI DevTool ecosystems. Sustainability must be a primary concern, and discussions about ethical innovation, software quality, and the future of digital systems should all include consideration of carbon debt. A collective action is then required for the transition to a carbon-aware digital economy.

To operationalise this concept, future research should focus on building measurement tools that can take into account both direct energy use (such as code completions) and indirect infrastructure emissions (such as CI/CD pipelines). By incorporating these measurements into developer environments through IDE plugins, the carbon debt of AI-assisted software development may become easier to understand and control. Additionally, carbon-aware DevTools that offer real-time feedback regarding carbon cost should be explored, and programmable "green modes" should be investigated that can restrict high-emission model invocation or promote lighter alternatives. In another view, research should be done on formalizing carbon debt as a measurable software quality attribute alongside performance and maintainability. Moreover, research should be done to

develop actual metrics for the suggested SIF, as no such standardized rating currently exists, and to test their applicability in workflows. This paper outlines an initial conceptual structure and indicators that shape the language and questions that empirical work must eventually address. This entails accounting for contextual factors such as infrastructure quality, energy systems, and developer behavior. Finally, more studies should be done on the ethical analysis of AI in Software Engineering beyond harm reduction to critically examine its necessity and bring about discussion on the non-adoption justification and/or minimalism of AI tools. The goal is to put sustainability at the center of responsible innovation debates.

REFERENCES

[1] The Guardian, "Elon Musk's xAI accused of pollution over Memphis supercomputer", [Retrieved: May 2025], Apr. 24, 2025, [Online]. Available: https://www.theguardian.com/technology/2025/apr/24/elon-musk-xai-memphis.

[2] L. Hampton *et al.*, "From Mining to E-waste: The Environmental and Climate Justice Implications of the Electronics Hardware Life Cycle", MIT Schwarzman College of Computing, 2024, [Online]. Available: https://mit-serc.pubpub.org/pub/w9ht6hue/release/5.

[3] J. Hess, "Chip Production's Ecological Footprint: Mapping Climate and Environmental Impact", [Retrieved: March 2025], 2024, [Online]. Available: https://www.interface-eu.org/publications/chip-productions-ecological-footprint.

[4] A. Zewe, "Explained: Generative AI's Environmental Impact", Published by MIT News on January 17, 2025, 2025, [Online]. Available: https://news.mit.edu/2025/explained-generative-ai-environmental-impact-0117.

[5] A. A. Fawole, O. F. Orikpete, N. N. Ehiobu, and D. R. E. Ewim, "Climate change implications of electronic waste: strategies for sustainable management", *Bulletin of the National Research Centre*, vol. 47, no. 1, p. 147, 2023.

[6] International Institute for Applied Systems Analysis, "Tracking net-zero carbon debt: Who is responsible for overshoot of the 1.5°C climate limit?", [Retrieved: April 2025], Mar. 2025, [Online]. Available: http://iiasa.ac.at/news/mar-2025/tracking-net-zero-carbon-debt (visited on 04/05/2025).

[7] W. Buchanan, *The Carbon Footprint Of AI*, https://devblogs.microsoft.com/sustainable-software/the-carbon-footprint-of-ai/, [Retrieved: April 2025], 2020.

[8] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, "Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162)", *Dagstuhl Reports*, vol. 6, no. 4, pp. 110–138, 2016, [Retrieved: April 2025].

[9] Y. Yu *et al.*, "Revisit the environmental impact of artificial intelligence: the overlooked carbon emission source?", *Frontiers of Environmental Science & Engineering*, vol. 18, no. 12, pp. 1–5, 2024.

[10] J. Vaidya and H. Asif, "A critical look at AI-generate software: Coding with the new AI tools is both irresistible and dangerous", *Ieee Spectrum*, vol. 60, no. 7, pp. 34–39, 2023.

[11] OECD, "Measuring the Environmental Impacts of Artificial Intelligence Compute and Applications: The AI Footprint", OECD Digital Economy Papers, Report 341, Nov. 2022, [Retrieved: 7 April 2025].

[12] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, *Green AI*, http://arxiv.org/abs/1907.1059, [Retrieved: April 2025], 2019. arXiv: 1907.1059.

[13] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for modern deep learning research", in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 13 693–13 696.

[14] A. De Vries, "The growing energy footprint of artificial intelligence", *Joule*, vol. 7, no. 10, pp. 2191–2194, 2023.

[15] R. Verdecchia, J. Sallou, and L. Cruz, "A Systematic Review of Green AI", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 13, no. 4, e1507, 2023.

[16] A. Singh, "Assessing the Carbon Footprint of OpenAI Models and Developing Strategies to Reduce It", Retrieved: April 2025, 2025, [Online]. Available: https://papers.ssrn.com/abstract=5195550.

[17] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, *Quantifying the Carbon Emissions of Machine Learning*, http://arxiv.org/abs/1910.09700, [Retrieved: June 2025], 2019.

[18] S. A. Budennyy *et al.*, "Eco2ai: Carbon emissions tracking of machine learning models as the first step towards sustainable AI", in *Doklady mathematics*, Springer, vol. 106, 2022, S118–S128.

[19] T. Zimmergren, *The Principles of Sustainable Software Engineering - Training*, https://learn.microsoft.com/en-us/training/modules/sustainable-software-engineering-overview/, [Retrieved: April 2025], 2025.

[20] D. Patterson *et al.*, "Carbon Emissions and Large Neural Network Training", *arXiv preprint arXiv:2104.10350*, 2021.

[21] GitHub Community, *How much CO_2/GHG does Copilot emit?*, https://github.com/orgs/community/discussions/38168, [Retrieved: April 2025], 2025.

[22] G. Fraser and A. Arcuri, "Evosuite: Automatic test suite generation for object-oriented software", in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, 2011, pp. 416–419.

[23] Diffblue, *Discover Diffblue Cover | Diffblue Documentation*, https://docs.diffblue.com, [Retrieved: April 2025], 2024.

[24] F. Palomba, A. Panichella, A. Zaidman, R. Oliveto, and A. De Lucia, "Automatic test case generation: What if test code quality matters?", in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, 2016, pp. 130–141.

[25] A. Hindle, "Green software engineering: The curse of methodology", in *2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER)*, IEEE, vol. 5, 2016, pp. 46–55.

[26] I. Manotas *et al.*, "An empirical study of practitioners' perspectives on green software engineering", in *Proceedings of the 38th international conference on software engineering*, 2016, pp. 237–248.

[27] Oxford University Press, *Debt Noun - Definition, pictures, pronunciation and usage notes*, https://www.oxfordlearnersdictionaries.com/definition/english/debt, [Retrieved: April 2025], 2025.

[28] K. Calvin *et al.*, *Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*, First, H. Lee and J. Romero, Eds. Geneva, Switzerland: Intergovernmental Panel on Climate Change (IPCC), Jul. 2023, [Retrieved 9 April 2025].

[29] Evans Data Corporation, *Worldwide Developer Population Report 24.2*, https://evansdata.com/reports/viewRelease.php?reportID=9, [Retrieved: April 2025], 2025.

[30] US Environmental Protection Agency (EPA), *Greenhouse Gas Equivalencies Calculator*, https://www.epa.gov/energy/greenhouse-gas-equivalencies-calculator, [Retrieved: 10 April 2025], 2015.

[31] US EPA, *Greenhouse Gas Emissions from a Typical Passenger Vehicle*, https://www.epa.gov/greenvehicles/greenhouse-gas-

emissions-typical-passenger-vehicle, [Retrieved: April 2025], 2016.

[32] Microsoft Corporation, *2022 Environmental Sustainability Report*, https://news.microsoft.com/wp-content/uploads/prod/sites/42/2023/05/2022-Environmental-Sustainability-Report.pdf, [Retrieved: April 2025], 2023.

[33] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, "Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions", *Communications of the ACM*, vol. 68, no. 2, pp. 96–105, 2025.

[34] B. Yetiştiren, I. Özsoy, M. Ayerdem, and E. Tüzün, "Evaluating the code quality of AI-assisted code generation tools: An empirical study on GitHub Copilot, Amazon Codewhisperer, and ChatGPT", *arXiv preprint arXiv:2304.10778*, 2023.

[35] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models", in *Chi conference on human factors in computing systems extended abstracts*, 2022, pp. 1–7.

[36] P. Henderson *et al.*, "Towards the systematic reporting of the energy and carbon footprints of machine learning", *Journal of Machine Learning Research*, vol. 21, no. 248, pp. 1–43, 2020.

[37] L. F. W. Anthony, B. Kanding, and R. Selvan, "Carbontracker: Tracking and predicting the carbon footprint of training deep learning models", *arXiv preprint arXiv:2007.03051*, 2020.

[38] K. Lottick, S. Susai, S. A. Friedler, and J. P. Wilson, "Energy Usage Reports: Environmental awareness as part of algorithmic accountability", *arXiv preprint arXiv:1911.08354*, 2019.

[39] N. Ding *et al.*, "Enhancing chat language models by scaling high-quality instructional conversations", *arXiv preprint arXiv:2305.14233*, 2023.

[40] P. Becker, *Sustainability science: Managing risk and resilience for sustainable development*. Elsevier, 2023.

[41] B. Penzenstadler *et al.*, "Everything is INTERRELATED: Teaching software engineering for sustainability", in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*, 2018, pp. 153–162.

[42] S. Lawrenz *et al.*, "Implementing the circular economy by tracing the sustainable impact", *International journal of environmental research and public health*, vol. 18, no. 21, p. 11 316, 2021.

[43] J. Fjeld, N. Achten, H. Hilligoss, A. Nagy, and M. Srikumar, "Principled artificial intelligence: Mapping consensus in ethical and rights-based approaches to principles for AI", *Berkman Klein Center Research Publication*, no. 2020-1, 2020.

[44] Z. Stein, "Technology Is Not Values Neutral: Ending the Reign of Nihilistic Design", *Consilience Project*, 2022.