

Dynamic Reasoning Budget Allocation for Efficient Large Language Model Inference

Arzu Wilmsen

Department of Computer Science, University of Central Florida, Orlando, FL, USA.

arzuwilmsen706@ucf.edu

Abstract

Large language models are increasingly deployed as general-purpose reasoning engines, yet their inference behavior remains largely governed by fixed computational schedules that ignore variation in task difficulty, user expectation, operational risk, and environmental constraints. This paper develops the concept of dynamic reasoning budget allocation as a system-level design principle for efficient and accountable large language model inference. Rather than treating reasoning depth as a static property of model architecture or prompt design, dynamic budgeting frames inference as a governed allocation problem in which computational effort is distributed across requests according to uncertainty, stakes, latency tolerance, and institutional policy. The discussion integrates perspectives from systems architecture, machine learning serving infrastructure, socio-technical governance, robustness evaluation, fairness, and sustainability. It argues that efficient inference is not merely a matter of reducing tokens or accelerating kernels, but of aligning expenditure of reasoning effort with the value and risk of each decision context. The paper examines architectural implications, including routing, caching, speculative execution, model cascades, and heterogeneous accelerator pools, and considers deployment challenges in cloud, edge, and regulated environments. It also analyzes fairness concerns arising when some users receive deeper reasoning than others, robustness risks from budget miscalibration, and policy questions concerning transparency, auditability, and environmental accounting. Case illustrations from customer support, scientific assistance, healthcare triage, and public-sector services show how dynamic budgeting changes system behavior across domains. The paper concludes that dynamic reasoning budget allocation offers a promising pathway toward scalable, equitable, and sustainable language model inference, but only if technical mechanisms are paired with explicit governance and continuous evaluation.

Keywords

dynamic reasoning budgets; large language model inference; systems architecture; sustainable computing; robustness; fairness; governance; deployment.

1. Introduction

Large language models have moved from research demonstrations to infrastructure-scale services that mediate writing, code generation, search, tutoring, customer support, scientific assistance, and administrative decision-making. Their capacity to perform multi-step reasoning has expanded through prompting strategies, reinforcement learning from human feedback, tool use, and inference-time scaling [1], [2], [3]. Yet the operational reality of serving these models is marked by a fundamental mismatch between uniform computational policies and highly variable cognitive demand. Many systems apply similar reasoning effort to trivial requests and high-stakes ambiguous problems, wasting computation on easy cases while under-serving difficult ones. This mismatch becomes more consequential as models are

embedded in workflows where latency, cost, energy consumption, reliability, and accountability are simultaneously important.

The idea of a reasoning budget refers to the amount of computational work allocated to a request before an answer is produced. In current practice, budgets are often implicit: a fixed maximum token count, a predetermined number of reasoning steps, a static retrieval depth, or a single model checkpoint selected by cost tier. Dynamic reasoning budget allocation makes this choice explicit and adaptive. It treats inference as a resource allocation problem in which the system decides how much deliberation a request deserves, when additional reasoning is likely to improve outcomes, and when further computation is wasteful or even harmful. This framing connects technical serving decisions to organizational policy, user expectations, and societal constraints.

The significance of this topic extends beyond latency optimization. Large language model inference has become a major component of the computational footprint of modern artificial intelligence, and its energy and financial costs are increasingly visible to operators, regulators, and publics [13], [14]. At the same time, fairness concerns arise when adaptive systems provide unequal reasoning quality to different users or groups, whether intentionally through pricing or unintentionally through biased routing. Robustness concerns arise when budget policies miscalibrate uncertainty, producing confident but shallow answers or unnecessarily expensive deliberation. Governance concerns arise when organizations cannot explain why a particular request received a particular level of computational effort. Dynamic reasoning budgeting therefore sits at the intersection of performance engineering and institutional accountability.

This paper argues that dynamic reasoning budget allocation should be understood as a socio-technical infrastructure capability rather than a narrow optimization trick. It requires architectural support for estimating task difficulty, policy support for deciding allocation rules, evaluation support for detecting failure modes, and economic support for aligning cost with value. The sections that follow examine foundations, architecture, deployment, governance, robustness, sustainability, case illustrations, and future research directions. The central claim is that efficient inference is not simply faster inference; it is inference whose reasoning effort is proportionally and transparently matched to the context in which it is used.

2. Foundations of Dynamic Reasoning Budgets

The conceptual foundation of dynamic budgeting lies in the recognition that reasoning is not monolithic. Some language model tasks require pattern completion, some require retrieval, some require multi-step deduction, and some require cautious abstention. Chain-of-thought prompting showed that intermediate reasoning steps can improve performance on complex tasks, while later work demonstrated that self-consistency and tool interaction can further enhance reliability [3], [4], [5]. However, these methods also increase computational cost and can introduce failure modes when reasoning traces become verbose without becoming more accurate. Dynamic budgeting responds by asking not whether reasoning is useful in general, but when and how much reasoning is useful for a specific request.

A second foundation is the distinction between uncertainty and difficulty. A request may be difficult for the model because it requires rare knowledge, ambiguous intent, or compositional inference. It may also be uncertain because the model lacks confidence, the input is noisy, or multiple answers are plausible. These conditions call for different responses. A difficult but well-specified task may benefit from deeper reasoning, retrieval, or tool use. An ambiguous

task may benefit from clarification before computation is spent. A high-confidence simple task may need no extra deliberation. Dynamic budgeting therefore depends on estimators that can distinguish these situations and policy rules that specify what to do when estimation is unreliable [6], [7].

A third foundation is the recognition that inference-time computation is a governed resource. In cloud environments, it consumes accelerator time, memory bandwidth, network capacity, and energy. In edge environments, it competes with battery life and local responsiveness. In regulated environments, it may need to be logged, justified, or capped for compliance. Treating reasoning as a budget makes these constraints visible and negotiable. It allows operators to define service classes, risk tiers, latency envelopes, and sustainability targets, and then to allocate reasoning effort accordingly. This is analogous to capacity planning in other critical infrastructures, where service-level objectives are balanced against reliability, cost, and public obligation [11], [12].

The foundations also include the limitations of static scaling. Simply increasing model size or context length does not guarantee better reasoning for every query, and it can worsen cost, latency, and environmental impact [8], [9]. Sparse expert models and conditional computation offer architectural precedents for spending computation selectively [10]. Dynamic reasoning budgeting extends this logic from parameter activation to inference-time deliberation. It suggests that a system can be efficient not only by activating fewer parameters, but by thinking less when thinking less is sufficient and thinking more when the stakes justify it.

3. Architectural Implications

Architecturally, dynamic reasoning budget allocation requires a control plane that observes request characteristics, estimates required effort, selects an execution path, and monitors outcome quality. This control plane may sit above model serving infrastructure and interact with routers, caches, retrieval systems, tool APIs, and safety filters. It may use lightweight classifiers, calibration models, semantic features, or historical performance data to estimate difficulty. The key architectural challenge is to make these decisions quickly enough that they do not erase the savings from reduced reasoning. If deciding how much to think costs nearly as much as thinking, the system gains little. Efficient estimators and cached policy decisions are therefore essential.

A second architectural implication is the need for heterogeneous execution paths. A dynamic system may choose among a small model, a medium model, a large model, a retrieval-augmented path, a tool-using path, or a multi-step reasoning path. It may also vary the number of sampled reasoning traces, the depth of search, or the degree of verification. Model cascades, speculative decoding, early-exit mechanisms, and mixture-of-experts routing all provide building blocks for such heterogeneity [10]. The architectural question is not which single mechanism is best, but how to compose them into a coherent policy that respects latency, cost, safety, and fairness constraints.

A third implication concerns memory and state. Dynamic budgeting often benefits from retaining intermediate reasoning artifacts, retrieval results, or tool outputs so that later stages can reuse them. This creates opportunities for caching and reuse across requests, especially in enterprise settings where similar questions recur. However, it also creates risks related to stale information, privacy leakage, and inconsistent reasoning across users. Systems must therefore define cache validity, access control, and invalidation policies. The design of memory hierarchy for reasoning is as important as the design of the reasoning policy itself.

A fourth implication is observability. Operators need to know why a request received a particular budget, how much computation was actually consumed, and whether the outcome justified the expenditure. This requires telemetry at the request level, including difficulty estimates, selected paths, token counts, tool calls, latency, cost, and user feedback. Without such observability, dynamic budgeting becomes opaque and difficult to govern. It also becomes difficult to improve, because the system cannot learn from cases where budgets were poorly matched to outcomes.

Recent work on scalable reasoning path filtering and alignment for small parameter models illustrates how budget-aware techniques can extend beyond frontier-scale systems [18]. By filtering reasoning paths and aligning outputs under resource constraints, such approaches suggest that dynamic budgeting can improve the viability of smaller models in environments where large-scale inference is impractical. This is particularly relevant for edge deployment, cost-sensitive applications, and organizations that need local control over data and computation. The architectural lesson is that budget allocation is not only a way to reduce cost at the top end, but also a way to make capable reasoning available under tighter resource envelopes.

4. Infrastructure and Deployment

Deployment contexts shape the feasibility and value of dynamic reasoning budgeting. In cloud data centers, operators can pool heterogeneous accelerators and use request-level scheduling to match budgets with available capacity. They can also implement tiered service levels, where premium requests receive deeper reasoning and best-effort requests receive lighter treatment. This raises fairness questions, because tiering may reproduce existing inequalities if essential services are available only to those who can pay. In edge and on-device settings, budgets are constrained by battery, thermal limits, memory, and intermittent connectivity. Dynamic budgeting can help by reserving deep reasoning for moments when it matters, while using lightweight responses for routine interactions.

Infrastructure also includes the networking and storage layers that support retrieval and tool use. A request that requires external knowledge may spend more time waiting for data than reasoning. Dynamic budgeting must therefore consider end-to-end latency rather than model computation alone. A system that allocates a large reasoning budget but then waits on a slow database may deliver worse user experience than a smaller budget with faster retrieval. This suggests that budget policies should be jointly optimized with data locality, caching, and prefetching strategies. It also suggests that infrastructure operators need to measure the tail of request latency, not just average performance, because adaptive policies can create unpredictable outliers when they misestimate difficulty [11].

Operational deployment also requires graceful degradation. If a budget estimator fails, if a tool becomes unavailable, or if load spikes, the system should fall back to safe and predictable behavior. In some contexts, fallback may mean using a smaller model with a warning. In others, it may mean asking the user to clarify or postponing the request. The design of fallback paths is a governance decision as much as an engineering one, because it determines who bears the cost of failure. A dynamic system that silently reduces reasoning quality for some users may be efficient but unfair; a system that always defaults to maximum reasoning may be robust but unsustainable.

Finally, deployment requires integration with existing organizational workflows. Dynamic budgeting cannot be evaluated in isolation from the decision processes it supports. In

customer support, a shallow answer may be acceptable if a human agent can intervene. In healthcare triage, a shallow answer may be dangerous if it discourages escalation. In scientific assistance, a deep but slow answer may be valuable if it improves reproducibility. Deployment success therefore depends on mapping budget policies to workflow roles, error tolerances, and accountability structures. This mapping is inherently interdisciplinary, drawing on systems engineering, human factors, and domain expertise.

5. Governance, Fairness, and Policy

Governance is central to dynamic reasoning budgeting because allocation decisions distribute a valuable resource: computational attention. When some requests receive more reasoning than others, the system is effectively making judgments about whose problems deserve deeper consideration. If these judgments are based only on willingness to pay, they may conflict with equitable access to public services, education, healthcare information, and legal assistance. If they are based on automated difficulty estimates that encode historical biases, they may systematically under-serve marginalized groups whose language, context, or needs are less represented in training data [15], [16]. Governance must therefore specify not only how budgets are allocated, but also what values guide the allocation.

Transparency is a related policy challenge. Users may not need to see every internal decision, but they may need to know whether they are receiving a best-effort answer, whether reasoning was limited by cost or policy, and whether they can request escalation. In regulated domains, auditors may need to reconstruct why a particular response was generated with a particular level of effort. This requires logs, versioning, and documentation of budget policies. It also requires clarity about responsibility when a limited budget contributes to an error. If an organization chooses to save computation by reducing reasoning on a high-stakes request, who is accountable for the consequences? Dynamic budgeting does not remove responsibility; it redistributes it across design, policy, and operation.

Fairness also involves the distribution of benefits and burdens. Efficient inference can reduce costs, making services more accessible. But if savings are captured by operators while users bear the risk of shallow answers, the result may be inequitable. Conversely, if deep reasoning is reserved only for exceptional cases, ordinary users may receive lower quality service without knowing it. A fair system might combine tiered service with minimum quality guarantees, transparent escalation paths, and independent monitoring of outcomes across groups. It might also require that essential public services meet a baseline reasoning standard regardless of ability to pay.

Policy implications extend to environmental accounting and procurement. Governments and institutions that purchase large language model services may need to consider not only price and accuracy but also energy intensity, carbon intensity, and the governance of reasoning budgets. Procurement rules could require disclosure of average and peak inference costs, sustainability metrics, and mechanisms for auditing allocation policies. International coordination may be necessary because models and data centers cross borders, but local values may differ regarding privacy, safety, and equity. Dynamic budgeting provides a technical lever that can support these policy goals, but it can also obscure them if allocation rules are hidden or unaccountable.

6. Robustness and Evaluation

Robustness in dynamic reasoning budgeting depends on the quality of difficulty estimation, the stability of allocation policies, and the ability to detect when a budget was insufficient or

excessive. A system may underestimate difficulty and produce a shallow answer that appears confident. This is especially dangerous in domains where users cannot easily verify correctness. Alternatively, it may overestimate difficulty and waste computation on simple requests, increasing cost and latency without improving outcomes. Evaluation must therefore measure both under-reasoning and over-reasoning, not only average accuracy or average cost.

Evaluation also faces the problem of counterfactuals. To know whether a larger budget would have improved an answer, the system must sometimes run additional reasoning, which defeats the purpose of saving computation. Offline evaluation can compare policies on historical data, but it may not capture distribution shift. Online evaluation can use randomized trials, but randomizing reasoning depth raises ethical questions when stakes are high. A practical approach is layered evaluation: shadow execution for low-risk requests, human review for high-risk domains, and continuous monitoring of proxies such as user corrections, escalation rates, and downstream errors. This aligns with broader lessons from machine learning systems about hidden technical debt and the need for monitoring beyond model accuracy [17].

Robustness also requires resilience to adversarial manipulation. If users learn that certain phrasings trigger deeper reasoning, they may game the system. If attackers can force excessive budgets, they may create denial-of-service conditions by inducing expensive deliberation. If attackers can suppress budgets, they may obtain shallow safety checks. Dynamic budgeting must therefore include rate limits, anomaly detection, and policy constraints that cannot be overridden by prompt content alone. The control plane must be treated as security-critical infrastructure, not as an optional optimization layer.

Finally, evaluation should include user trust and comprehension. If users cannot tell whether an answer was produced with limited reasoning, they may over-trust it. If they are told too often that reasoning was limited, they may lose confidence in the system. The design of explanations and escalation mechanisms is therefore part of robustness. A robust dynamic budgeting system is not one that always chooses the correct budget, but one that communicates uncertainty, provides recourse, and learns from errors without hiding them.

7. Sustainability and Economic Trade-offs

The sustainability case for dynamic reasoning budgeting is straightforward in principle: if many requests do not require deep reasoning, allocating less computation to them reduces energy use, cooling demand, hardware wear, and financial cost. Prior work on energy and policy considerations for deep learning has shown that computational choices have material environmental consequences [14]. More recent analyses of large model training and deployment have emphasized the importance of measuring and managing carbon emissions across the lifecycle [13]. Dynamic budgeting extends this concern to inference, which may dominate operational energy as models are used at scale.

However, sustainability is not only about reducing average computation. It is also about avoiding rebound effects. If efficient inference makes language models cheaper, demand may increase, potentially offsetting savings. If dynamic budgeting enables deployment on smaller devices, it may expand the total number of active models. If operators use saved capacity to serve more requests rather than to reduce environmental impact, the net effect may be ambiguous. Sustainability governance must therefore connect efficiency gains to explicit targets, such as carbon caps, renewable energy matching, or absolute reduction commitments.

Economically, dynamic budgeting creates new questions about pricing and value. Should users pay for tokens, for reasoning depth, for outcomes, or for service-level guarantees? A

system that charges by tokens may incentivize shallow answers. A system that charges by outcome may be difficult to implement when value is uncertain. A system that charges by budget tier may be transparent but potentially inequitable. The economic design must align incentives so that operators are rewarded for appropriate allocation rather than for maximum consumption or minimum cost at the expense of quality.

There are also trade-offs between centralization and distribution. Large cloud providers can implement sophisticated dynamic budgeting because they have scale, telemetry, and heterogeneous hardware. Smaller organizations may lack these capabilities, leading to dependence on external platforms. Open standards for budget policies, telemetry formats, and evaluation metrics could help level the playing field. Public infrastructure for shared evaluation and auditing could also support smaller actors. Without such measures, dynamic budgeting may become another source of concentration in the artificial intelligence economy.

8. Case Illustrations and Cross-Domain Comparisons

Customer support provides a clear illustration of dynamic budgeting. Many user inquiries are routine and can be answered with a small model, cached knowledge, or a scripted response. Others involve billing disputes, safety concerns, or complex troubleshooting that benefit from deeper reasoning and tool use. A dynamic system can route simple requests to fast paths and escalate difficult ones to more expensive reasoning or human agents. The key governance question is how escalation is triggered and whether users can request it. If escalation is hidden, users may receive inconsistent service without understanding why.

Scientific assistance offers a different pattern. Researchers may ask for literature summaries, hypothesis generation, code debugging, or statistical interpretation. Some tasks require broad retrieval and careful synthesis, while others require quick formatting or translation. In this domain, deep reasoning may be valuable even when latency is high, because the cost of error can be significant. Dynamic budgeting might allow users to specify a reasoning preference, such as speed, balance, or thoroughness, while the system manages underlying resources. This resembles high-performance computing job scheduling, where users request quality-of-service levels and the scheduler allocates resources accordingly.

Healthcare triage illustrates the risks of under-reasoning. A request that appears simple may have hidden clinical complexity. If a dynamic system allocates a small budget based on surface features, it may miss dangerous symptoms or rare interactions. In such settings, minimum reasoning budgets, mandatory uncertainty disclosure, and human oversight may be necessary. Dynamic budgeting can still help by directing extra computation to ambiguous or high-risk cases, but it cannot replace clinical governance. The domain shows that efficiency must be bounded by safety constraints that are set externally, not only by optimization objectives.

Public-sector services, including benefits eligibility, immigration guidance, and legal aid, raise fairness concerns. If deep reasoning is reserved for users who can pay or who appear more articulate, the system may worsen existing inequalities. A fair design might guarantee a minimum budget for all users, allow free escalation for high-stakes questions, and audit allocation patterns across demographic groups. Cross-domain comparison suggests that dynamic budgeting is not a single universal policy but a family of policies whose parameters must reflect domain-specific values, risk tolerances, and accountability structures [19], [20].

9. Research Agenda and Forward-Looking Perspectives

A first research priority is the development of reliable and interpretable difficulty estimators. These estimators should predict not only whether additional reasoning will improve accuracy, but also whether it will improve safety, fairness, and user satisfaction. They should be calibrated under distribution shift and auditable by domain experts. Research should also examine how estimators can be trained without collecting sensitive user data or creating feedback loops that advantage already well-served populations.

A second priority is the design of policy languages and governance interfaces for budget allocation. Organizations need ways to express rules such as minimum reasoning for medical questions, maximum cost for internal drafting, or preferential escalation for vulnerable users. These rules should be composable, testable, and transparent. They should also be integrated with existing compliance and risk management frameworks. Without such interfaces, dynamic budgeting will remain an ad hoc engineering practice rather than a governable infrastructure capability.

A third priority is evaluation infrastructure. The field needs benchmarks that measure the relationship between reasoning budget and outcome quality across domains, including rare and high-stakes cases. It also needs methods for auditing allocation decisions without exposing private data. Shared evaluation platforms could allow researchers, regulators, and civil society to compare policies and identify harms. Such infrastructure is costly, but it is necessary if dynamic budgeting is to be trusted at scale.

A fourth priority is sustainable and equitable deployment. Research should examine how dynamic budgeting interacts with hardware efficiency, renewable energy availability, and geographic load balancing. It should also study how pricing, subsidies, and public procurement can ensure that efficiency gains do not simply widen access gaps. The long-term vision is a reasoning infrastructure that treats computational attention as a resource to be allocated with care, not merely consumed. This vision connects technical optimization to democratic accountability and environmental responsibility.

10. Conclusion

Dynamic reasoning budget allocation offers a powerful framework for efficient large language model inference because it aligns computational effort with the actual demands of each request. It moves beyond static token limits and uniform reasoning policies toward adaptive systems that can distinguish routine tasks from complex ones, low-stakes interactions from high-stakes decisions, and well-understood queries from ambiguous ones. Architecturally, it requires control planes, heterogeneous execution paths, memory management, observability, and fallback mechanisms. Operationally, it requires integration with cloud, edge, and regulated environments. Governance-wise, it requires attention to fairness, transparency, accountability, and environmental impact.

The central argument of this paper is that efficiency in language model inference is not only a technical property but a socio-technical achievement. A system that reduces computation without considering who receives deeper reasoning, how errors are handled, or what environmental costs remain may be efficient in a narrow sense but unjust or fragile in a broader sense. Conversely, a system that invests in governance, evaluation, and equitable access can use dynamic budgeting to make capable reasoning more sustainable and more widely available.

Future work should therefore treat dynamic reasoning budgets as a first-class design object. Researchers should develop better estimators, policy interfaces, evaluation benchmarks, and

auditing mechanisms. Operators should document allocation rules and provide escalation paths. Policymakers should consider minimum quality guarantees, environmental accounting, and transparency requirements. If these efforts are pursued together, dynamic reasoning budget allocation can help shape an inference infrastructure that is not only faster and cheaper, but also more robust, fair, and accountable.

References

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998–6008.
2. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901.
3. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35, 24824–24837.
4. Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35, 22199–22213.
5. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2023). Self-consistency improves chain of thought reasoning in language models. *International Conference on Learning Representations*.
6. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. *International Conference on Learning Representations*.
7. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36.
8. Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., ... Sifre, L. (2022). Training compute-optimal large language models. *Advances in Neural Information Processing Systems*, 35, 30016–30030.
9. Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.
10. Fedus, W., Zoph, B., & Shazeer, N. (2022). Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120), 1–39.
11. Dean, J., & Barroso, L. A. (2013). The tail at scale. *Communications of the ACM*, 56(2), 74–80.

12. Barroso, L. A., & Hölzle, U. (2009). The datacenter as a computer: An introduction to the design of warehouse-scale machines. Morgan & Claypool.
13. Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L. M., Rothchild, D., So, D., Texier, M., & Dean, J. (2021). Carbon emissions and large neural network training. arXiv preprint arXiv:2104.10350.
14. Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 3645–3650.
15. Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, 610–623.
16. Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., ... Liang, P. (2021). On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258.
17. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. Advances in Neural Information Processing Systems, 28, 2503–2511.
18. Li, Q. (2026, May). TrajLite PPO: Scalable Reasoning Path Filtering and Alignment for Small Parameter Large Language Models. In 2026 2nd International Conference on Artificial Intelligence and Digital Ethics (ICAIDE) (pp. 29-32). IEEE.
19. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askill, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 35, 27730–27744.
20. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. arXiv preprint arXiv:1606.06565.