

Adaptive Sparse Computation for Resource-Efficient Graph Learning

Gerlind Hanel

School of Computing, Clemson University, Clemson, SC, USA.
gerlindhanel02@clemson.edu

Leonie A. MacGuinness

Department of Computer Science, University of Houston, Houston, TX, USA.
leonie.a.macguinness@uh.edu

Abstract

Graph learning has become a foundational capability for modeling relational data in social networks, scientific discovery, transportation, finance, and public infrastructure. Yet the computational intensity of graph neural networks creates persistent tensions among accuracy, latency, energy use, and equitable access to machine learning services. Adaptive sparse computation offers a system-level response by dynamically reducing redundant or low-value work across graph structure, node features, message passing, and training trajectories. This paper examines adaptive sparsity not as a single algorithmic trick but as an architectural and governance paradigm. It analyzes structural trade-offs among memory locality, irregular access, communication overhead, and model fidelity, while connecting those trade-offs to hardware kernels, graph partitioning, runtime scheduling, and deployment environments. The discussion extends to federated and edge infrastructures, sustainability accounting, robustness under distribution shift, fairness risks created by uneven sparsification, and policy implications for auditability and resource accountability. Case illustrations from social computing, molecular discovery, mobility analytics, and healthcare collaboration show that resource-efficient graph learning requires coordinated decisions across data representation, compiler support, hardware design, and institutional oversight. The paper concludes that adaptive sparse computation can improve resource efficiency and broaden participation in graph learning, but only when sparsity decisions are evaluated as socio-technical choices rather than purely local optimizations. Without careful governance, sparsity can amplify existing inequalities, obscure failure modes, and shift hidden costs to users and environments.

Keywords

adaptive sparsity, graph neural networks, resource-efficient learning, graph systems, sustainable computing, fairness, federated learning, system governance.

1. Introduction

Graph learning has moved from a specialized research area to a core component of contemporary artificial intelligence infrastructure. Early graph neural network models established that relational structure can be incorporated into representation learning through neighborhood aggregation and message passing [4]. Subsequent architectures such as GraphSAGE, graph convolutional networks, and graph attention networks demonstrated that inductive learning, spectral-inspired convolution, and attention over neighbors can support node classification, link prediction, and graph-level prediction at scale [1], [2], [3]. Surveys of the field now document a broad ecosystem of methods spanning recommendation, chemistry,

biology, knowledge representation, and traffic forecasting [5], [6]. This expansion has produced significant computational pressure. Real graphs are often large, skewed, heterogeneous, and continuously changing, making training and inference expensive in ways that differ from dense image or sequence workloads.

The central systems problem is that graph learning combines irregular memory access with high arithmetic demand. Message passing repeatedly gathers features from neighbors, transforms them, and scatters updates, producing workloads whose efficiency depends on graph topology, feature dimensionality, batching strategy, and hardware memory hierarchy. Dense neural network optimizations often transfer poorly because graph neighborhoods are uneven and dynamic. As a result, practitioners face a difficult structural trade-off: increasing model capacity and neighborhood depth can improve predictive quality, but it also increases memory traffic, synchronization, and energy consumption. Adaptive sparse computation responds to this tension by treating sparsity as a runtime and architectural resource rather than a fixed property of the model. It asks which edges, features, activations, or gradient pathways can be omitted at a given moment without unacceptable loss in utility, and how those omissions interact with system bottlenecks.

Resource-efficient graph learning is therefore not merely a matter of compressing tensors. It involves decisions about representation, scheduling, partitioning, kernel design, deployment locus, and accountability. A sparse graph algorithm may reduce arithmetic operations while worsening memory locality. A pruning policy may lower storage and bandwidth but introduce fairness disparities if frequently sampled subgraphs systematically exclude minority communities. A runtime system may accelerate training on a graphics processing unit but increase carbon intensity if it runs in a region with dirty energy. These concerns place adaptive sparse computation at the intersection of computer architecture, machine learning, infrastructure policy, and social responsibility. This paper develops an integrated account of that intersection and argues that adaptive sparsity should be evaluated through system-level metrics that include accuracy, latency, energy, robustness, fairness, and auditability.

2. Foundations and System Semantics of Sparsity

Sparsity in graph learning appears in several distinct forms, and each carries different system consequences. Structural sparsity refers to missing edges or deliberately pruned neighborhoods. Feature sparsity refers to absent, zero, or low-information node attributes. Activation sparsity refers to hidden representations that contribute little to downstream computation. Gradient sparsity refers to updates that can be skipped or compressed during training. Temporal sparsity refers to the fact that many graphs change slowly, so not every node or edge requires recomputation at every step. These forms are related but not interchangeable. A system that exploits activation sparsity may reduce arithmetic operations while leaving irregular memory access unchanged, whereas a system that exploits structural sparsity may reduce memory footprint but create load imbalance across partitions.

Deep learning research has established that pruning and compression can drastically reduce model size and inference cost [7]. The lottery ticket hypothesis further suggested that some sparse subnetworks can retain competitive accuracy when trained from suitable initialization [8]. Subsequent analyses clarified that pruning outcomes depend heavily on architecture, sparsity pattern, retraining schedule, and evaluation protocol [9]. Broader surveys of sparsity in deep learning describe a design space that includes structured and unstructured pruning, growth strategies, quantization, and hardware-aware sparsification [10]. Graph learning inherits these insights but must adapt them to relational structure. Unlike dense layers, graph

operations derive meaning from neighborhoods; removing an edge can remove information about social ties, molecular bonds, or infrastructure dependencies. Sparsity is therefore not only a compression mechanism but also a representational intervention.

Graph-specific sparsification methods illustrate this dual character. DropEdge regularization randomly removes edges during training to improve generalization and reduce over-smoothing [11]. Sampling-based methods such as FastGCN use importance sampling to approximate neighborhood aggregation more efficiently [12]. These approaches show that sparsity can serve both computational and statistical purposes. However, they also expose a fundamental trade-off: stochastic sparsification may reduce variance in dense neighborhoods but increase variance in rare or small neighborhoods. In a social network, a highly connected hub may tolerate aggressive sampling, while a peripheral user may be represented poorly if the few available edges are pruned. Thus, the semantics of sparsity must be evaluated not only by global accuracy but also by distributional effects across graph regions, communities, and task-relevant subgroups.

From a system perspective, sparsity is meaningful only when it changes resource consumption in a measurable way. Zero values in memory still consume bandwidth if they are loaded and processed. Sparse data structures such as compressed formats can reduce storage but may complicate vectorization and increase indexing overhead. Adaptive sparse computation therefore requires co-design among data layout, compiler transformations, runtime schedulers, and hardware capabilities. The most effective systems do not simply remove values; they reorganize computation so that skipped work translates into lower memory traffic, fewer synchronization events, or reduced energy use. This distinction between nominal sparsity and effective sparsity is central to resource-efficient graph learning.

3. Architectural and Kernel-Level Considerations

At the architectural level, graph learning is constrained by memory behavior more than by raw arithmetic throughput in many practical settings. Neighborhood aggregation involves gathering features from non-contiguous locations, applying transformations, and scattering results. If the graph is large, feature caches may not fit on chip, and random accesses can dominate latency. Kernel-level designs for graph neural network training have therefore focused on improving memory coalescing, reducing redundant loads, and scheduling computation around high-degree vertices. Recent work on fast graphics processing unit kernel design for graph neural network training demonstrates how careful control of work assignment and memory movement can substantially improve throughput [13]. This line of research illustrates that sparsity must be aligned with hardware execution patterns; otherwise, fewer operations may still be slower because of poor utilization.

Software frameworks provide another layer of architectural mediation. Libraries such as PyTorch Geometric and the Deep Graph Library offer abstractions for batching, message passing, and heterogeneous graph operations [14], [15]. These frameworks allow researchers to express sparse computation, but their performance depends on partitioning, memory allocation, and kernel fusion. Scalable training systems have explored redesigned dataflows that reduce intermediate materialization and improve locality across layers [16]. Adaptive runtime systems further adjust execution plans according to graph characteristics, hardware state, and workload phase [17]. Such systems reveal that sparsity decisions are inseparable from scheduling. A sparse neighborhood may be efficient if its vertices are grouped by locality, but inefficient if they are scattered across partitions.

The trade-offs are architectural rather than incidental. Structured sparsity can improve vectorization and load balance but may remove more information than necessary. Unstructured sparsity preserves fine-grained relevance but often delivers limited speedup on general-purpose hardware. Dynamic sparsity can adapt to changing graph activity but introduces control overhead and unpredictable latency. Partitioning strategies that reduce cross-device communication may create imbalanced partitions, while balanced partitions may generate expensive remote transfers. In federated settings, communication sparsification can reduce bandwidth but may slow convergence if important updates are consistently omitted. These tensions mean that adaptive sparse computation cannot be reduced to a single optimization objective. It requires multi-objective design across accuracy, throughput, energy, robustness, and fairness.

Kernel and compiler support also affects reproducibility and portability. A sparsity policy tuned for one graphics processing unit may not transfer to another, to central processing units, or to edge accelerators. If implementations rely on vendor-specific primitives, deployment becomes brittle and auditing becomes difficult. A more sustainable architecture separates policy from mechanism: the system expresses which computations may be sparsified and under what service-level constraints, while backend compilers and kernels choose concrete layouts and schedules. This separation allows the same adaptive sparsity logic to operate across cloud clusters, edge devices, and heterogeneous accelerators. It also enables governance because sparsity decisions can be logged as explicit policy events rather than hidden implementation details.

4. Adaptive Control and Runtime Governance

Adaptive sparse computation becomes genuinely adaptive when it uses feedback to adjust sparsity over time. A static pruning mask is determined once and then applied uniformly. An adaptive system observes signals such as loss trends, gradient norms, memory pressure, latency budgets, energy prices, or distribution shift indicators, and then changes sparsity levels accordingly. In graph learning, useful signals include neighborhood size, vertex degree, edge freshness, feature entropy, and partition-level congestion. Runtime systems can increase sparsity when the system is resource-constrained and relax it when accuracy or robustness is at risk. This control loop must be governed carefully because sparsity decisions can create path dependence: once certain edges or features are repeatedly omitted, the model may fail to learn their importance.

The governance challenge is that adaptive sparsity often operates below the level of user visibility. A service may report high average accuracy while systematically underperforming on a subgraph or community. To address this, runtime governance should maintain auditable sparsity policies, including thresholds, sampling rates, and override conditions. Explainability research warns that explanations can be misleading if they do not account for the full computational path that produced a prediction [23]. In a sparse graph system, an explanation that references retained edges but ignores pruned ones may be incomplete. Auditors need to know which structural evidence was available to the model at inference time and which was discarded before computation. This is especially important in high-stakes domains such as credit, public benefits, healthcare, and criminal justice, where relational data can encode sensitive associations.

Governance also involves service-level objectives beyond accuracy. A resource-efficient graph learning service might promise maximum latency, energy per query, or bandwidth consumption. Adaptive sparsity can help meet these objectives, but it can also game them by

degrading rare cases first. For example, a system under load might sparsify neighborhoods with low connectivity because they are cheaper to drop. If those neighborhoods correspond to marginalized users, the service may meet its average latency target while deepening inequity. A robust governance framework should therefore combine aggregate performance metrics with subgroup and tail metrics. It should also define escalation paths when sparsity-induced uncertainty exceeds tolerable bounds, such as falling back to denser computation for contested decisions or requesting human review.

Federated and collaborative graph learning intensify these governance questions. In federated settings, clients may hold sensitive local graphs, and communication sparsification can reduce bandwidth and improve privacy by transmitting fewer updates [20]. However, sparsification can also bias the global model toward clients with dense, well-connected data. Clients with sparse local graphs may contribute fewer or noisier updates, causing their needs to be underrepresented. Adaptive sparsity policies should therefore account for client heterogeneity, not only server-side efficiency. This requires coordination mechanisms that allow clients to declare resource constraints and fairness requirements without revealing sensitive graph structure. Such mechanisms are as much institutional as technical, because they determine who has voice in setting sparsity thresholds and who bears the cost of approximation.

5. Deployment, Infrastructure, Sustainability, and Policy

Deployment location shapes the sustainability and equity profile of graph learning. Cloud data centers offer scalable accelerators and centralized management but consume significant energy and water for cooling. Edge deployment can reduce wide-area communication and support latency-sensitive applications, but edge devices have limited memory and power. Adaptive sparsity can help bridge these environments by matching computation to available resources. A cloud service might use aggressive sparsification for low-priority batch analytics, while an edge device might use conservative sparsification for safety-critical local inference. The same graph model can thus operate under different sparsity regimes depending on infrastructure constraints and policy priorities.

Sustainability accounting must go beyond nominal operation counts. Energy use depends on hardware generation, utilization, cooling efficiency, and the carbon intensity of the local grid. Green artificial intelligence research argues for reporting energy and carbon alongside accuracy [21], [22]. For graph learning, this means evaluating whether sparse computation actually reduces total energy rather than merely shifting work to indexing or communication. A sparse kernel that runs longer due to irregular memory access may consume more energy than a dense kernel. A distributed sparse system may reduce per-device computation but increase network traffic and synchronization energy. Policy makers and infrastructure operators should therefore require life-cycle accounting that includes training, inference, data movement, and hardware embodied emissions.

Policy implications extend to procurement and access. If resource-efficient graph learning is deployed only on advanced accelerators, smaller organizations and public institutions may be excluded from participating. Adaptive sparsity can lower barriers if it enables useful graph learning on commodity hardware, but it can also create a two-tier ecosystem in which well-resourced actors use dense, high-fidelity computation while others receive sparse approximations. Public policy can encourage open benchmarks that report accuracy, energy, latency, and subgroup performance together. Funding agencies can support shared infrastructure for reproducible sparsity research. Regulators can require disclosure when

automated decisions rely on sparsified relational evidence, especially when those decisions affect rights, opportunities, or access to essential services.

Infrastructure design should also consider resilience. A system that depends on a single accelerator type or a single cloud region is vulnerable to supply chain disruption, outages, and price shocks. Adaptive sparse computation can improve resilience by allowing workloads to degrade gracefully across heterogeneous hardware. If a high-end accelerator becomes unavailable, a sparsity-aware scheduler can reduce neighborhood depth or feature dimensionality to maintain service. However, graceful degradation must be bounded by policy. Some applications should fail safely rather than produce low-confidence sparse predictions. The design question is not whether sparsity is always beneficial, but when it is acceptable, who decides, and how the decision is communicated to affected parties.

6. Robustness, Fairness, and Auditability

Robustness and fairness are tightly coupled in adaptive sparse graph learning. Sparsity changes the effective receptive field of each node, which can alter sensitivity to adversarial edges, noisy features, and distribution shift. A model trained with aggressive edge sampling may be robust to missing links but brittle when important rare edges are present. Conversely, a model trained with dense neighborhoods may overfit to spurious connections. Adaptive systems that change sparsity at inference time can introduce additional instability because the model encounters computational conditions not seen during training. Robustness testing should therefore include sparsity variation as a dimension of stress testing, not merely a performance optimization.

Fairness risks arise when sparsity interacts with graph topology in unequal ways. Social networks, citation graphs, and infrastructure networks often exhibit power-law degree distributions. High-degree nodes have many redundant paths, so losing a few edges may have little effect. Low-degree nodes may depend on a small number of edges, so the same sparsity rate can be devastating. If an adaptive policy allocates computation according to degree, it may systematically underserve peripheral nodes. If it allocates computation according to historical activity, it may reinforce existing inequalities. Fairness-aware sparsity should therefore include topology-aware metrics such as coverage of low-degree nodes, representation of minority communities, and error rates on structurally disadvantaged subgraphs. These metrics should be monitored during training and deployment.

Auditability is necessary for both robustness and fairness. An auditable sparse graph system should record which nodes, edges, and features were retained or omitted for a given decision, within the limits of privacy and security. It should distinguish between deliberate sparsification, natural missingness, and data collection bias. It should also document the policy that governed sparsity, including who set it, what objective it served, and what fallback mechanisms existed. Such records support incident investigation, regulatory review, and public accountability. Without them, sparse computation becomes an opaque layer that can absorb responsibility for errors that actually stem from system design choices.

Cross-domain comparisons clarify why context matters. In molecular property prediction, sparsity may reflect chemical bonds, and omitting an edge can change scientific meaning [18]. In recommendation, sparsity may reduce latency but risk filter bubbles if weak ties are consistently pruned. In transportation, sparse computation can help process large mobility graphs, but fairness concerns arise if certain neighborhoods receive less accurate routing or demand prediction. In healthcare collaboration, federated graph learning can protect patient

data, yet sparsity-induced underrepresentation of small clinics may harm model quality for rare conditions [20]. Computational social science reminds us that relational data are often traces of human behavior, not neutral measurements [24]. Adaptive sparsity therefore inherits the ethical complexities of the data and institutions in which it is deployed.

7. Case Illustrations and Forward-Looking Perspectives

A useful case illustration is a municipal service platform that uses graph learning to predict infrastructure failures. The graph connects roads, utilities, sensors, and maintenance records. During peak load, an adaptive sparse system might sample sensor neighborhoods and prune stale maintenance edges to meet latency budgets. This can reduce cost and improve responsiveness. However, if sparsity systematically removes edges from historically underinvested neighborhoods, the system may underestimate risk in those areas. A responsible deployment would combine sparsity with equity constraints, ensuring that sampling rates are not lower in vulnerable regions and that maintenance predictions are audited across neighborhoods. It would also integrate human review for high-impact decisions and disclose the role of automated relational inference.

Another case is a scientific collaboration network used to recommend potential research partnerships. Adaptive sparsity can reduce the computational burden of scoring millions of possible links. Yet if the system prunes weak or cross-disciplinary ties, it may reinforce disciplinary silos and reduce novelty. Here the objective should balance efficiency with exploratory value. A sparsity policy might deliberately preserve a quota of weak ties or cross-community edges, treating them as exploration resources rather than noise. This illustrates a broader principle: sparse computation should be aligned with the epistemic goals of the application, not only with hardware constraints. Efficiency is valuable when it expands capability, but harmful when it narrows the space of possibilities in ways that reproduce existing power structures.

Forward-looking research should pursue several directions. First, sparsity-aware compilers and runtime systems should expose policy hooks that allow operators to specify fairness, robustness, and energy constraints. Second, benchmarks should report not only accuracy and speed but also subgroup error, tail latency, energy per inference, and carbon intensity. Third, hardware designs should support structured and dynamic sparsity without sacrificing programmability. Fourth, federated and edge systems should develop privacy-preserving mechanisms for negotiating sparsity across heterogeneous clients. Fifth, governance frameworks should clarify liability when sparse approximations contribute to harm. These directions require collaboration among machine learning researchers, computer architects, social scientists, legal scholars, and community stakeholders.

The long-term promise of adaptive sparse computation is that it can make graph learning more accessible, sustainable, and resilient. Smaller organizations could run useful relational models on modest hardware. Energy-constrained edge devices could perform local inference without constant cloud communication. Large-scale scientific graphs could be analyzed with lower carbon cost. However, that promise depends on treating sparsity as a governed system property. If sparsity is optimized only for throughput, it may externalize costs onto underrepresented users, obscure failure modes, and increase long-term technical debt. If it is optimized within a multi-objective framework, it can support both efficiency and democratic accountability.

8. Conclusion

Adaptive sparse computation offers a powerful systems strategy for resource-efficient graph learning, but its value depends on how it is integrated with architecture, infrastructure, and governance. Graph learning workloads are irregular, memory-intensive, and socially embedded. Sparsity can reduce computation, communication, and energy, yet it can also alter representational meaning, exacerbate topological inequality, and complicate auditability. The most promising path is not maximum sparsity at any cost. It is adaptive, policy-aware sparsity that responds to runtime conditions while respecting accuracy, robustness, fairness, and sustainability constraints. This requires co-design across kernels, frameworks, schedulers, deployment environments, and institutions. It also requires transparent reporting of what is omitted, why it is omitted, and who is affected. By treating sparse computation as a socio-technical choice rather than a purely local optimization, the field can advance graph learning systems that are more efficient, more resilient, and more equitable in their distribution of benefits and burdens.

References

1. Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems* 30 (pp. 1024–1034). Curran Associates.
2. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.
3. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.
4. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1), 61–80.
5. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1), 4–24.
6. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1, 57–81.
7. Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *International Conference on Learning Representations*.
8. Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.
9. Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., & Gutttag, J. (2020). What is the state of neural network pruning? In *Proceedings of Machine Learning and Systems* 2 (pp. 129–146).
10. Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., & Peste, A. (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241), 1–124.
11. Rong, Y., Huang, W., Xu, T., & Huang, J. (2020). DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*.

12. Chen, J., Ma, T., & Xiao, C. (2018). FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*.
13. Peng, H., Xie, X., Shivdikar, K., Hasan, M. A., Zhao, J., Huang, S., ... & Ding, C. (2024, April). Maxk-gnn: Extremely fast gpu kernel design for accelerating graph neural networks training. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (pp. 683-698).
14. Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
15. Wang, M., Yu, L., Zheng, D., Gan, Q., Gai, Y., Ye, Z., Li, C., Zhou, J., Huang, Q., Ma, C., Huang, Z., Guo, Q., Zhang, H., Lin, H., Zhao, J., Li, J., Smola, A. J., & Zhang, Z. (2019). Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.
16. Jia, Z., Lin, S., Ying, R., You, J., Leskovec, J., & Aiken, A. (2020). Redesigning the graph convolutional network dataflow for scalable training. In *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (pp. 212–225).
17. Wang, Y., Feng, B., Li, G., Yang, S., Xiong, Y., Xie, Y., & Ding, C. (2021). GNNAdvisor: An adaptive and efficient runtime system for GNN acceleration on GPUs. In *2021 USENIX Annual Technical Conference* (pp. 515–531).
18. Zhang, M., Cui, Z., Neumann, M., & Chen, Y. (2018). An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 4438–4445.
19. Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? In *International Conference on Learning Representations*.
20. Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... & Zhao, S. (2021). Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2), 1–210.
21. Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 3645–3650).
22. Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2020). Green AI. *Communications of the ACM*, 63(12), 54–63.
23. Mittelstadt, B., Russell, C., & Wachter, S. (2019). Explaining explanations in AI. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (pp. 279–288).
24. Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabási, A. L., Brewer, D., ... & Van Alstyne, M. (2009). Computational social science. *Science*, 323(5915), 721–723.