

Sparse Matrix Processing for High-Performance Graph Neural Networks

Olimpia Sobrino

Department of Computer Science, University of New Hampshire, Durham, NH, USA.
olimpiamail@unh.edu

Tuukka Kivisto

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

Robin Mccullough

Department of Computer Science, George Mason University, Fairfax, VA, USA.
mccullough1990@gmu.edu

Abstract

Graph neural networks have become a central computational paradigm for learning over relational, networked, and irregularly structured data. Their performance depends heavily on sparse matrix processing, because message passing, neighborhood aggregation, and feature transformation are commonly expressed as sparse matrix operations. This paper examines sparse matrix processing for high-performance graph neural networks from a system-level perspective rather than treating it as a narrow kernel optimization problem. It analyzes how sparse representations, memory hierarchy behavior, parallel execution, distributed partitioning, deployment infrastructure, sustainability, robustness, fairness, and governance interact in real-world graph learning systems. The discussion emphasizes structural trade-offs: formats that improve locality may increase preprocessing costs, compression that reduces memory traffic may complicate vectorization, and distributed partitioning that improves scalability may worsen communication or weaken privacy guarantees. The paper also considers how hardware acceleration, including graphics processing units and specialized sparse kernels, affects the broader design space. It argues that high-performance graph neural network systems require coordinated decisions across algorithms, compilers, runtimes, data infrastructure, and policy. Performance is therefore not merely a matter of throughput but also of reproducibility, energy use, fairness, auditability, and long-term maintainability. The conclusion outlines open challenges in adaptive sparse formats, hardware-software co-design, distributed fairness, sustainable deployment, and governance frameworks for graph-based artificial intelligence.

Keywords

sparse matrix processing, graph neural networks, high-performance computing, GPU acceleration, distributed systems, sustainability, fairness, infrastructure governance

1 Introduction Graph neural networks have emerged as a powerful class of machine learning models for domains in which entities and relationships are naturally represented as graphs. Early formulations of graph neural networks established the idea that node states can be learned through iterative message passing over graph structure [1]. Subsequent work demonstrated that graph convolutional networks can perform semi-supervised node classification by aggregating information from local neighborhoods [2], while inductive

approaches enabled generalization to unseen nodes and graphs [3]. Attention-based graph models further enriched the expressive capacity of neighborhood aggregation [4], and theoretical analyses clarified the limits and strengths of message-passing architectures [5]. Across these developments, a common computational pattern appears: information is gathered from neighboring nodes, transformed, and combined, often repeatedly across layers. When implemented efficiently, this pattern is frequently mapped onto sparse matrix operations, especially sparse matrix-dense matrix multiplication and sparse matrix-sparse matrix multiplication. The system-level importance of sparse matrix processing follows directly from the irregularity of real graphs. Social networks, citation networks, recommendation graphs, biological networks, and transportation networks often exhibit highly skewed degree distributions. A small number of nodes may have enormous neighborhoods, while many nodes have very few connections. This irregularity creates load imbalance, poor memory locality, and unpredictable access patterns. General surveys of graph neural networks have documented the breadth of architectures and application domains [6], and reviews have emphasized both methodological progress and practical bottlenecks [7]. However, the performance of graph neural networks is not determined by model architecture alone. It depends on how sparse structures are stored, how kernels are scheduled, how memory bandwidth is used, how graphs are partitioned across devices, and how systems are deployed in cloud, edge, or hybrid environments. This paper therefore treats sparse matrix processing as an infrastructural concern that connects algorithms, hardware, data governance, and sustainability. The central argument is that high-performance graph neural networks require a holistic design stance. A sparse format that is optimal for one graph may be inefficient for another. A GPU kernel that accelerates training on a single device may not scale well when communication dominates. A distributed training pipeline that improves throughput may introduce privacy risks or complicate fairness auditing. A compression technique that reduces energy per operation may increase preprocessing overhead or reduce reproducibility. These trade-offs are not incidental; they are constitutive of the system. The sections that follow examine sparse representations, hardware architecture, distributed deployment, governance, robustness, fairness, sustainability, evaluation practices, and future directions.

2 Background and Related Work

Graph neural networks typically operate through repeated neighborhood aggregation. In many implementations, adjacency information and node features are combined through sparse matrix operations. This formulation is attractive because mature sparse linear algebra libraries offer optimized routines for sparse matrix-vector multiplication, sparse matrix-dense matrix multiplication, and related kernels. The graph neural network model introduced by Scarselli and colleagues [1] provided an early foundation, while graph convolutional networks [2] and inductive representation learning [3] popularized practical training pipelines. Graph attention networks [4] introduced learned weighting of neighbors, and subsequent theoretical work examined expressive power and limitations [5]. Surveys and reviews have synthesized these developments [6], [7], but they also highlight that performance engineering remains a major obstacle. Hardware acceleration has become a major research direction. A recent GPU kernel design for accelerating graph neural network training [8] illustrates how aggressive kernel-level innovation can improve throughput for sparse and irregular workloads. Such work is important because graphics processing units offer high memory bandwidth and massive parallelism, yet they are often underutilized by sparse graph operations. Frameworks such as PyTorch Geometric [9] and Deep Graph Library [10] have lowered the barrier to implementing graph neural networks by providing sparse data structures, sampling routines, and optimized message-passing primitives. Contrastive and self-supervised graph learning methods [11] further increase the diversity of workloads,

because they may require multiple views, negative sampling, or large batches. Benchmark suites such as the Open Graph Benchmark [12] have improved comparability by standardizing datasets and evaluation protocols. Sparse matrix research provides a deeper foundation. The University of Florida sparse matrix collection [13] has long served as a reference for evaluating sparse kernels across scientific computing domains. Sparse matrix-vector multiplication on throughput-oriented processors [14] established many principles for GPU sparse computation, including the importance of coalesced access and load balancing. Merge-based parallel sparse matrix-vector multiplication [15] advanced these ideas by improving workload distribution. More recent work on sparse matrix multiplication on GPUs [16] has explored data transformation and execution strategies that reduce divergence and improve memory efficiency. Beyond single-device acceleration, multi-GPU graph processing systems [17] and distributed graph neural network training platforms [18] address scaling across devices and nodes. Comprehensive graph neural network platforms such as AliGraph [19] show that industrial deployment requires ingestion, sampling, partitioning, training, and serving to be integrated. At the infrastructure layer, the warehouse-scale computing perspective [20] helps explain why graph learning systems must be understood as parts of larger data center ecosystems.

3 Sparse Matrix Representations and Graph Neural Network Workloads

Sparse matrix representations determine much of the performance envelope for graph neural networks. The compressed sparse row format stores nonzeros by row and is often effective for row-wise aggregation, but it can suffer when rows have highly variable lengths. The coordinate format is flexible and convenient for construction, yet it may require additional sorting or conversion before efficient computation. Blocked and hybrid formats can improve locality for graphs with clustered structure, but they may introduce overhead for extremely irregular graphs. The choice of format influences memory consumption, preprocessing time, kernel selection, and portability across hardware. Research on sparse matrix collections [13], sparse matrix-vector multiplication [14], [15], and sparse matrix multiplication [16] has repeatedly shown that no single representation dominates all workloads. Graph neural network workloads amplify these concerns because they combine sparse and dense operations. Neighborhood aggregation is sparse, but feature transformation is often dense or mixed. A single layer may involve sparse matrix multiplication with a feature matrix, followed by dense linear transformation, normalization, and activation. During training, backward passes introduce additional sparse operations and gradient accumulation. Mini-batch sampling can reduce the size of sparse operations but may introduce sampling overhead and variance. Full-graph training can improve gradient quality but may exceed memory capacity. These trade-offs connect directly to sparse matrix processing because they determine the shape, density, and locality of the matrices being manipulated. Benchmark datasets such as those in the Open Graph Benchmark [12] and framework implementations in PyTorch Geometric [9] and Deep Graph Library [10] provide practical contexts for evaluating these choices. Irregularity is the defining challenge. In many real graphs, degree distributions are heavy-tailed. A kernel that assigns equal work to each row will therefore be imbalanced. A memory access pattern that assumes consecutive neighbors may stall when neighbors are scattered. A partitioning strategy that minimizes edge cuts may still produce highly uneven partitions in terms of vertex degrees. Attention-based models [4] and expressive message-passing models [5] can further increase computational cost because they may compute pairwise attention or more complex aggregation. The result is that sparse matrix processing for graph neural networks cannot be reduced to a single optimized routine. It requires a portfolio of representations, scheduling policies, and runtime decisions that adapt to graph structure and training phase.

4 Hardware and System Architecture

Graphics processing units

are widely used for graph neural network training because they provide high throughput for dense operations and increasingly sophisticated support for sparse computation. However, sparse graph workloads often underutilize them due to warp divergence, uncoalesced memory access, and insufficient parallelism in small neighborhoods. A recent GPU kernel design for accelerating graph neural network training [8] demonstrates that careful kernel engineering can substantially improve performance, but such optimizations must be integrated with memory management, data movement, and framework-level scheduling. Sparse matrix-vector multiplication research [14], [15] offers lessons about balancing work across threads and exploiting hierarchical memory. Sparse matrix multiplication strategies [16] show that data transformation and execution order can be as important as the arithmetic itself. The memory hierarchy is a central architectural constraint. Sparse matrices and feature matrices compete for cache, shared memory, and high-bandwidth memory. If a sparse format stores indices inefficiently, memory traffic increases. If feature dimensions are large, dense feature blocks may dominate capacity. If many small kernels are launched, launch overhead and synchronization costs can become significant. On modern accelerators, specialized matrix units and sparse tensor capabilities may help, but they often require structured sparsity patterns that do not naturally arise in graphs. Consequently, system architects must decide whether to transform graph data into hardware-friendly structures, accept lower utilization, or design custom kernels that tolerate irregularity. This is a classic trade-off between generality and peak performance. Scaling beyond a single device introduces additional architectural concerns. Multi-GPU graph processing systems [17] must partition graphs, replicate features, and manage communication. Distributed graph neural network training platforms [18] face similar challenges at cluster scale, including sampling, synchronization, and fault tolerance. Industrial graph platforms such as AliGraph [19] emphasize end-to-end integration, because training performance depends on data ingestion and storage as much as on kernel speed. The warehouse-scale computing perspective [20] reminds us that these systems operate within data centers where power, cooling, network topology, and failure rates shape feasible designs. Thus, hardware architecture for high-performance graph neural networks is inseparable from deployment context and infrastructure governance.

5 Distributed Deployment and Infrastructure Governance

Distributed deployment is often necessary for large graphs that cannot fit on one device. Partitioning strategies determine communication volume, load balance, and memory footprint. Edge-cut partitioning may reduce cross-partition edges, while vertex-cut partitioning may better handle high-degree vertices. Sampling can reduce the working set, but it changes the statistical properties of training and may require careful tuning. Distributed graph neural network training systems [18] have developed sophisticated pipelines for sampling, feature fetching, and gradient synchronization. Multi-GPU graph processing systems [17] show that communication and memory management can dominate runtime. Comprehensive platforms [19] integrate these concerns with storage, scheduling, and serving. Infrastructure governance becomes critical when graph data crosses organizational boundaries. Graphs often encode sensitive relationships: friendships, transactions, medical interactions, or communication patterns. Distributed training may replicate features across nodes, create intermediate caches, or log sampled subgraphs. Each of these steps raises questions about data minimization, access control, retention, and auditability. The data center as a computer [20] provides a useful lens because it highlights how resource management, isolation, and failure recovery are systemic properties. Governance frameworks must therefore consider not only whether a model is accurate but also whether the infrastructure that produces it respects legal, ethical, and organizational constraints. This is especially important in regulated domains such as finance, healthcare, and public services. Deployment

environments also vary. Cloud deployment offers elasticity and centralized management, but it may introduce data transfer costs and jurisdictional issues. Edge deployment can reduce latency and keep data local, but it constrains memory, power, and connectivity. Hybrid architectures may split training and inference, or perform sampling at the edge and aggregation in the cloud. Each pattern changes the sparse matrix processing workload. Edge devices may favor compressed formats and small batches, while cloud clusters may favor high-throughput kernels and large partitions. A governance-aware design must align technical choices with policy requirements, including privacy, security, and fairness auditing. Without such alignment, high-performance graph learning can scale efficiently while still producing socially problematic outcomes.

6 Robustness, Fairness, and Policy Implications Sparse matrix processing affects robustness because approximations and optimizations can change model behavior. Pruning edges, dropping neighbors, quantizing features, or sampling subgraphs may improve speed but also alter the information available to the model. Adversarial attacks on graph structure can exploit these approximations, especially when defense mechanisms rely on sparsity assumptions. Theoretical work on expressive power [5] and contrastive graph learning [11] highlights that graph representations are sensitive to structural perturbations. Robustness therefore requires testing across sparsity levels, degree distributions, and adversarial conditions. A kernel that performs well on a benchmark graph may behave poorly on a noisy or adversarial graph, and the resulting model may be less reliable in deployment. Fairness is also entangled with sparse processing. If graph sampling or partitioning systematically underrepresents certain communities or relation types, the resulting model may produce biased predictions. A survey on bias and fairness in machine learning [23] provides a broad framework for understanding these risks. In graph settings, bias can arise from missing edges, homophily, unequal feature availability, or uneven neighborhood sizes. Sparse optimizations can exacerbate these issues if they disproportionately discard minority-group connections or reduce the effective receptive field for underrepresented nodes. Model reporting practices such as model cards [24] can improve transparency by documenting intended use, performance disparities, and evaluation conditions. However, model cards alone are insufficient if the underlying infrastructure cannot trace how sparse transformations affected training data. Policy implications follow from these technical realities. Regulators and organizations may require explanations for automated decisions, audits of training data, or assessments of disparate impact. A graph neural network system that uses aggressive sparse approximation may be difficult to audit because the effective training graph differs from the raw graph. Governance frameworks should therefore require documentation of sparse transformations, partitioning choices, sampling strategies, and hardware-specific optimizations. They should also encourage reproducible benchmarks that include fairness and robustness metrics, not only accuracy and throughput. In this sense, sparse matrix processing is not merely an engineering detail; it is a site where technical choices become policy-relevant.

7 Sustainability and Lifecycle Sustainability is an increasingly important system-level concern. Deep learning has significant energy costs, and natural language processing research has shown that model training can produce substantial carbon emissions [21]. Green artificial intelligence advocates argue for reporting energy use and efficiency alongside accuracy [22]. Graph neural networks are not exempt from these concerns. Sparse operations can reduce computation by avoiding zero entries, but they can also increase memory traffic and irregular execution. A sparse kernel may consume less arithmetic energy while using more energy for data movement, indexing, and synchronization. The net effect depends on hardware, graph structure, and implementation quality. Sustainable design therefore requires measuring energy per useful operation, not merely speed. The lifecycle of hardware and infrastructure matters as

well. Data centers consume energy for computation, cooling, and networking, and they have embodied carbon costs from construction and equipment manufacturing. The warehouse-scale computing perspective [20] emphasizes that efficiency improvements at the chip level may be offset by higher-level inefficiencies in scheduling, replication, or data movement. Graph learning systems often maintain multiple copies of graphs and features, run repeated experiments, and store checkpoints. These practices can increase storage and network energy use. Sustainable deployment should consider the full lifecycle: data ingestion, preprocessing, training, inference, serving, and retirement. Sparse matrix processing can contribute to sustainability, but only if it is integrated with system-wide energy accounting. There is also a rebound effect to consider. When graph neural networks become faster and cheaper, practitioners may train larger models on larger graphs more frequently. Efficiency gains can therefore increase total energy consumption rather than reduce it. Policy interventions such as carbon-aware scheduling, energy reporting, and efficiency standards may be necessary to align performance optimization with sustainability goals. In this context, sparse matrix processing is both a technical opportunity and a governance challenge. It can reduce the cost of graph learning, but it can also enable unsustainable scale if left unmanaged.

8 Evaluation and Case Illustrations

Evaluation practices shape the direction of research and deployment. Benchmark suites such as the Open Graph Benchmark [12] provide standardized datasets and tasks, while sparse matrix collections [13] offer diverse structural patterns for kernel evaluation. Framework ecosystems like PyTorch Geometric [9] and Deep Graph Library [10] enable rapid experimentation, but they may abstract away important performance details. Distributed platforms such as DistDGL [18] and AliGraph [19] illustrate how production systems integrate sampling, partitioning, training, and serving. A recent GPU kernel design [8] shows the potential of low-level optimization, while sparse matrix multiplication research [16] demonstrates the value of data transformation and execution strategy. Together, these works suggest that evaluation should be multi-dimensional: accuracy, latency, throughput, memory, energy, fairness, robustness, and maintainability. Consider a social network recommendation scenario. The graph is large, heterogeneous, and dynamic. Training may use neighbor sampling to control memory, but sampling can bias the model toward active users. Sparse matrix kernels must handle high-degree nodes without load imbalance. Distributed training may shard the graph across devices, introducing communication overhead. If the system uses aggressive compression, it may reduce network traffic but complicate auditability. If it uses fairness-aware sampling, it may improve representation but reduce throughput. These trade-offs cannot be resolved by a single metric. They require a governance process that weighs business objectives, user rights, and operational constraints. Consider a scientific application such as drug discovery. Graphs may represent molecules or protein interactions. Sparse matrix processing is essential because chemical graphs are irregular and often small but numerous. Batching many small graphs creates new challenges: padding wastes memory, while ragged batching complicates kernel design. Robustness is critical because incorrect predictions may have safety implications. Fairness may concern representation across disease populations or chemical classes. Sustainability may favor efficient models that can run on modest hardware. Evaluation should therefore include domain-specific validation, uncertainty quantification, and reproducibility. The same sparse processing decisions that improve speed may affect scientific validity, so they must be documented and justified.

9 Open Challenges and Future Directions

Several open challenges remain. First, adaptive sparse formats are needed. Static formats cannot accommodate the diversity of graph structures and training phases. Runtime systems could select or synthesize formats based on graph statistics, hardware capabilities, and layer type. Second, hardware-software co-design must continue.

GPU kernel innovations [8] and sparse matrix multiplication strategies [16] show that significant gains are possible, but they often require deep integration with compilers and frameworks. Third, distributed systems must address fairness and privacy alongside scalability. DistDGL [18] and AliGraph [19] provide foundations, but governance mechanisms for cross-organizational graph learning remain immature. Fourth, sustainability must become a first-class objective. Green AI [22] and energy-aware deep learning research [21] offer principles, but graph-specific energy models are underdeveloped. Fifth, robustness and fairness require better evaluation. Fairness surveys [23] and model reporting frameworks [24] can guide practice, but they must be adapted to graph structure and sparse processing. Sixth, policy frameworks must keep pace with technical change. Data protection, antitrust, and non-discrimination laws may all apply to graph-based artificial intelligence. Sparse matrix processing can obscure how data flows through a system, making audits more difficult. Future work should therefore develop transparency mechanisms that expose sparse transformations without revealing sensitive data. Finally, the field needs shared abstractions. Graph neural network frameworks [9], [10] have improved productivity, but performance portability remains elusive. A kernel optimized for one GPU may not transfer to another. A partitioning strategy that works for one graph may fail for another. A fairness intervention that helps one group may harm another. The challenge is to build systems that are adaptive, auditable, and sustainable without sacrificing the performance gains that make graph neural networks practical. This requires collaboration across computer architecture, machine learning, distributed systems, social science, and policy.

10 Conclusion Sparse matrix processing is a foundational concern for high-performance graph neural networks. It influences how graphs are stored, how kernels execute, how systems scale, and how models behave under real-world constraints. This paper has argued that sparse matrix processing should be understood as a system-level discipline rather than a narrow optimization task. Representation choices interact with memory hierarchy, hardware acceleration, distributed partitioning, deployment infrastructure, governance, robustness, fairness, and sustainability. A recent GPU kernel design for accelerating graph neural network training [8] demonstrates the potential of low-level innovation, but such innovation must be embedded in a broader architecture of accountability and measurement. The path forward requires adaptive formats, hardware-software co-design, distributed fairness, energy-aware deployment, and policy frameworks that make graph learning systems transparent and responsible. High-performance graph neural networks will not be defined by speed alone, but by their ability to operate efficiently, equitably, and sustainably within complex socio-technical infrastructures.

References

1. 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. <https://doi.org/10.1109/TNN.2008.2005605>
2. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.
3. Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems* (pp. 1024–1034).
4. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

5. Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? In International Conference on Learning Representations.
6. 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. <https://doi.org/10.1109/TNNLS.2020.2978386>
7. 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. <https://doi.org/10.1016/j.aiopen.2021.01.001>
8. 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).
9. Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
10. Wang, M., Yu, L., Zheng, D., Gan, Q., Gai, Y., Ye, Z., ... & Li, Y. (2019). Deep Graph Library: Towards efficient and scalable deep learning on graphs. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
11. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., & Wang, L. (2020). Deep graph contrastive representation learning. *arXiv*. <https://arxiv.org/abs/2006.04131>
12. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., & Leskovec, J. (2020). Open Graph Benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems* (Vol. 33, pp. 22118–22133).
13. Davis, T. A., & Hu, Y. (2011). The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1), 1–25. <https://doi.org/10.1145/2049662.2049663>
14. Bell, N., & Garland, M. (2009). Implementing sparse matrix-vector multiplication on throughput-oriented processors. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (pp. 1–11). <https://doi.org/10.1145/1654059.1654078>
15. Merrill, D., & Garland, M. (2016). Merge-based parallel sparse matrix-vector multiplication. In *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis* (pp. 678–689). IEEE. <https://doi.org/10.1109/SC.2016.57>
16. Jiang, P., Hong, C., & Agrawal, G. (2020). A novel data transformation and execution strategy for accelerating sparse matrix multiplication on GPUs. In *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (pp. 376–388). <https://doi.org/10.1145/3373376.3378493>
17. Jia, Z., Kwon, Y., Shipman, G., McCormick, P., Erez, M., & Aiken, A. (2018). A distributed multi-GPU system for fast graph processing. *Proceedings of the VLDB Endowment*, 11(3), 297–310. <https://doi.org/10.14778/3157794.3157799>

18. Zheng, D., Ma, C., Wang, M., Zhou, J., Su, Q., Wang, X., ... & Wang, Y. (2020). DistDGL: Distributed graph neural network training for billion-scale graphs. In 2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3) (pp. 36–44). IEEE. <https://doi.org/10.1109/IA351965.2020.00011>
19. Zhu, R., Zhao, K., Yang, H., Lin, W., Zhou, C., Ai, B., Li, Y., & Zhou, J. (2019). AliGraph: A comprehensive graph neural network platform. *Proceedings of the VLDB Endowment*, 12(12), 2094–2105. <https://doi.org/10.14778/3352063.3352127>
20. Barroso, L. A., Hölzle, U., & Ranganathan, P. (2018). *The datacenter as a computer: Designing warehouse-scale machines* (3rd ed.). Morgan & Claypool. <https://doi.org/10.2200/S00874ED3V01Y201809CAC046>
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). <https://doi.org/10.18653/v1/P19-1355>
22. Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2020). Green AI. *Communications of the ACM*, 63(12), 54–63. <https://doi.org/10.1145/3381831>
23. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6), 1–35. <https://doi.org/10.1145/3457607>
24. Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., ... & Gebru, T. (2019). Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (pp. 220–229). <https://doi.org/10.1145/3287560.3287596>