FPGA-Based Acceleration of Graph Convolutional Networks for Edge Intelligence

Mick Ketting
Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS, USA.
mketting@ku.edu

Nayeli Rascon
Department of Computer Science and Engineering, University of Nevada, Reno, Reno, NV, USA.
contactnayeli@unr.edu

Raksha Bal
Department of Computer Science, Colorado State University, Fort Collins, CO, USA.
contactraksha@colostate.edu

Abstract

Graph convolutional networks have become a central computational method for learning from relational data, yet their deployment in edge intelligence settings remains constrained by irregular memory access, high energy costs, and tight latency budgets. This paper examines field-programmable gate array acceleration of graph convolutional networks as a system-level strategy for edge intelligence rather than as a narrow hardware optimization problem. It reviews the architectural characteristics of graph convolution, including neighborhood aggregation, sparse traversal, and dynamic graph structure, and connects these characteristics to the constraints of edge infrastructures. The analysis considers trade-offs among reconfigurable logic, hardened processors, graphics processing units, and application-specific integrated circuits, with attention to quantization, dataflow design, memory hierarchy, graph partitioning, and runtime orchestration. The paper also examines governance, deployment, sustainability, robustness, and fairness implications that arise when graph learning is embedded into socio-technical infrastructures such as smart cities, industrial systems, and healthcare platforms. It argues that FPGA-based acceleration can provide a compelling balance of energy efficiency, adaptability, and latency control, but only when hardware design is coupled with careful data governance, lifecycle management, and fairness auditing. The conclusion identifies open challenges in programming productivity, dynamic graph support, security, benchmarking, and interdisciplinary evaluation, and proposes that future research treat edge graph intelligence as an integrated infrastructure problem involving silicon, software, institutions, and policy.

Keywords

FPGA acceleration; graph convolutional networks; edge intelligence; heterogeneous computing; socio-technical infrastructure; robustness; fairness; governance.

1. Introduction

Graph convolutional networks have emerged as a powerful class of machine learning models for relational data, including social networks, molecular structures, transportation systems,

citation graphs, and sensor networks. Early formulations of graph neural networks established the idea that node representations can be learned through iterative message passing over graph structure [5]. Subsequent work on graph convolutional networks demonstrated that localized spectral approximations can support semi-supervised classification with strong empirical performance [1]. Inductive extensions such as GraphSAGE broadened the approach to unseen nodes and large-scale dynamic settings [2], while attention-based aggregation and expressive message-passing variants further expanded the design space [3], [4]. Survey work has since consolidated methodological advances and application patterns across domains [6], [7]. The result is a mature modeling paradigm whose computational demands, however, remain poorly matched to conventional edge hardware.

Edge intelligence refers to the deployment of machine learning capabilities near data sources, devices, gateways, and local infrastructure rather than exclusively in centralized cloud data centers. This shift is motivated by latency, bandwidth, privacy, energy, and autonomy concerns [20], [21]. For graph convolutional networks, edge deployment is especially challenging because graph workloads combine sparse memory access, irregular neighborhood sizes, dynamic topology, and repeated aggregation. Unlike dense image or language models, graph models often exhibit low arithmetic intensity, poor locality, and unpredictable communication. Graphics processing units can deliver high throughput for some graph workloads, and specialized accelerator research has explored GPU and application-specific designs [8], [9], [10], [14]. Nevertheless, edge environments impose constraints that make reconfigurable logic attractive. Field-programmable gate arrays can be tailored to sparse graph traversal, mixed precision arithmetic, and streaming aggregation while maintaining lower power envelopes than many general-purpose accelerators [15], [16], [17].

This paper argues that FPGA-based acceleration of graph convolutional networks should be understood as an integrated system problem. The relevant questions are not only whether a kernel can be mapped to programmable logic, but how that mapping interacts with edge infrastructure, update cycles, governance, robustness, sustainability, and fairness. A design that achieves high throughput on a static benchmark may fail when deployed in a municipal sensor network, a hospital gateway, or an industrial control loop. Conversely, a modest accelerator that supports secure updates, predictable latency, and transparent auditing may produce greater long-term value. The following sections examine graph convolution workloads, architectural trade-offs, FPGA acceleration strategies, deployment and governance, robustness and sustainability, fairness, cross-domain cases, and open challenges.

2. Graph Convolutional Workloads and Edge Constraints

Graph convolutional networks operate by propagating information across graph edges and combining it with node features. In the simplest formulation, each node aggregates representations from its neighbors and transforms the result through learned weights [1]. This computation is conceptually simple but structurally difficult. Neighborhoods may vary from a few edges to millions, adjacency matrices are often sparse, and access patterns depend on graph topology. Real-world graphs also change: social interactions evolve, transportation networks experience disruptions, and sensor topologies lose or gain nodes. Inductive methods such as GraphSAGE were developed partly to handle such settings by learning aggregation functions that generalize to unseen nodes [2]. Attention mechanisms can weight neighbors differently [3], and expressive message-passing architectures can distinguish graph structures that simpler aggregators cannot [4]. These modeling choices influence hardware requirements

because attention, edge features, and multi-hop sampling increase memory traffic and control complexity.

Edge constraints amplify these difficulties. Edge devices often have limited dynamic random-access memory, constrained thermal budgets, intermittent connectivity, and heterogeneous processors. Cloud offloading can reduce local computation but introduces network latency, bandwidth cost, and privacy exposure [20], [21]. For latency-critical applications such as collision avoidance, industrial anomaly detection, or emergency triage, offloading may be unacceptable. For privacy-sensitive applications such as patient similarity graphs or personal recommendation, offloading may also raise regulatory and ethical concerns. Edge intelligence therefore seeks a balance among local computation, selective communication, and centralized coordination. FPGA acceleration fits this balance because programmable logic can be reconfigured for different graph operators, precision levels, and interface requirements without designing a new application-specific chip for every deployment.

However, graph convolutional networks are not uniformly suited to FPGA acceleration. Their irregularity complicates pipelining, their sparsity can cause underutilization, and their dynamic structure challenges static scheduling. Accelerator surveys emphasize that graph neural network performance depends heavily on dataflow, memory hierarchy, and graph partitioning [8], [11]. Hardware acceleration research has explored both general frameworks and specialized architectures [12], [13]. The challenge for edge intelligence is to preserve enough flexibility for changing graphs and model versions while still exploiting the energy efficiency of custom data paths. This tension between specialization and adaptability is the central architectural theme of FPGA-based graph acceleration.

3. Architectural Landscape and Trade-Offs

The architectural landscape for graph convolutional acceleration includes central processing units, graphics processing units, application-specific integrated circuits, and field-programmable gate arrays. Graphics processing units offer mature programming ecosystems and high parallelism, and recent work has proposed high-performance GPU accelerator designs for graph convolution networks [14]. They are often well suited to batched operations and dense subgraphs, but their energy consumption and reliance on external memory can be problematic at the edge. Application-specific integrated circuits can achieve exceptional efficiency for fixed operators, but they lack post-deployment flexibility and require long design cycles. Field-programmable gate arrays occupy a middle ground: they offer spatial parallelism and custom memory organization while remaining reconfigurable after deployment [15], [16]. This reconfigurability is valuable when graph schemas, model quantizations, or regulatory requirements change.

The trade-offs are multidimensional. First, throughput versus latency: edge applications may require bounded latency rather than maximum average throughput. A design that batches many graphs to improve utilization may violate real-time constraints. Second, energy versus precision: reduced precision can lower energy and memory traffic, but graph aggregation can be sensitive to quantization error, especially when node features have wide dynamic range [17], [19]. Third, flexibility versus efficiency: a highly specialized pipeline for one aggregation function may be efficient but unable to support attention or edge-conditioned convolution. Fourth, on-chip memory versus external bandwidth: FPGAs can use block RAM and distributed memory for local aggregation, but large graphs require external memory and careful partitioning. Fifth, static scheduling versus dynamic runtime: precompiled dataflows can reduce overhead, but dynamic graphs demand runtime adaptation [13].

These trade-offs are also governance trade-offs. Reconfigurable hardware can be updated remotely, which supports bug fixes and model improvements, but remote update mechanisms create supply-chain and cybersecurity risks. A compromised bitstream or model update could alter inference behavior without obvious software symptoms. Similarly, energy-efficient designs may encourage more pervasive sensing, which raises surveillance and consent concerns. A system-level analysis must therefore consider not only performance per watt but also accountability, verifiability, and reversibility. The most efficient accelerator is not necessarily the most legitimate or sustainable deployment.

4. FPGA Acceleration Strategies for Graph Convolutional Networks

FPGA acceleration strategies for graph convolutional networks generally address dataflow, memory access, quantization, and graph partitioning. Dataflow architectures map graph operations onto processing elements that stream node features and edge lists through aggregation and transformation stages. Because graph traversal is irregular, designers often use decoupled access-execute patterns, scratchpad memories, and reordering techniques to improve locality. Pipelining can overlap memory reads with arithmetic, while spatial parallelism can replicate aggregation units for different partitions. Frameworks such as FINN and FINN-R illustrate how reconfigurable dataflow designs can support quantized neural network inference with high efficiency [15], [16]. Although these frameworks focus on dense neural networks, their lessons about quantization, streaming, and resource balancing apply to graph operators as well.

Graph partitioning is especially important. Large graphs must be divided into subgraphs that fit on-chip memory or that can be processed with limited external communication. Partitioning quality affects load balance, communication volume, and latency. For edge deployments, partitioning may also reflect physical topology: sensors in one building, vehicles in one corridor, or patients in one clinic may form natural subgraphs. This creates an opportunity for topology-aware acceleration, where hardware scheduling follows the structure of the edge network. However, it also creates fairness and privacy risks, because partitions can encode social or geographic segregation. A partition that improves locality may also isolate vulnerable communities from shared resources.

Quantization and mixed precision are central to FPGA efficiency. Reduced precision reduces memory bandwidth and logic usage, enabling more parallelism within a fixed power budget [18], [19]. Yet graph convolution can accumulate errors across hops, and attention weights may require finer resolution. Adaptive precision, where sensitive nodes or edges use higher precision, can balance accuracy and energy. Such adaptation requires runtime monitoring and may complicate verification. Sparsity exploitation is similarly double-edged. Skipping zero edges saves computation, but irregular sparsity can cause branch divergence and memory inefficiency. Reordering and blocking can mitigate this, but they add preprocessing cost. For dynamic graphs, preprocessing may need to run continually, creating a trade-off between accelerator efficiency and system-level overhead.

Runtime systems and compilers are therefore as important as the accelerator fabric. GNNAdvisor, for example, explores adaptive runtime support for graph neural network acceleration on GPUs [13]. Similar principles apply to FPGA deployments: the runtime must manage graph loading, partitioning, precision selection, and scheduling across heterogeneous resources. Hardware description languages and high-level synthesis can improve productivity, but they may not expose enough control for aggressive optimization. The most promising direction is a co-designed stack in which graph algorithms, compiler transformations, and

FPGA data paths are developed together. Without this co-design, FPGA graph acceleration risks becoming a collection of impressive but brittle kernels.

5. Edge Deployment, Infrastructure, and Governance

Deploying FPGA-accelerated graph convolutional networks at the edge requires infrastructure that spans devices, gateways, network links, orchestration services, and cloud backends. Cloudlets and edge computing architectures provide useful conceptual foundations for placing computation near users while retaining centralized management where appropriate [20], [21]. In practice, an edge graph intelligence system might include sensors that generate graph updates, local FPGAs that perform neighborhood aggregation, gateways that coordinate model versions, and a cloud service that trains or audits models. Each layer introduces governance questions. Who owns the graph data? Who can update the model? How are bitstreams signed and verified? What telemetry is collected, and for how long? These questions are not peripheral to hardware design; they determine whether the system can be trusted and maintained.

Infrastructure governance also affects performance. Remote attestation, secure boot, and encrypted communication consume resources and may require dedicated logic. Version control for FPGA bitstreams and model weights must be synchronized with software containers and data schemas. If a gateway updates its accelerator but another node retains an older graph schema, inference results may diverge. In safety-critical settings, such divergence is unacceptable. Governance therefore needs mechanisms for staged rollouts, rollback, and heterogeneous compatibility. These mechanisms resemble familiar software deployment practices, but they must account for the fact that hardware configuration is part of the application. A corrupted bitstream is not merely a software bug; it can change the physical behavior of the device.

Policy implications follow. Edge graph intelligence can support public-interest applications such as traffic optimization, disaster response, and preventive healthcare. It can also enable intrusive profiling if deployed without oversight. Regulatory frameworks for data protection, algorithmic accountability, and critical infrastructure security must therefore intersect with accelerator design. For example, privacy-preserving aggregation can be implemented in hardware, but its guarantees depend on how graphs are constructed and how updates are shared. Fairness auditing requires access to model behavior and data provenance, which may conflict with proprietary bitstreams or trade secrets. A legitimate deployment must make enough information available for independent evaluation while protecting sensitive data and intellectual property. This balance is inherently political and institutional, not only technical.

6. Robustness, Sustainability, and Fairness

Robustness in FPGA-accelerated graph intelligence has multiple dimensions. Hardware robustness includes resilience to soft errors, thermal throttling, voltage variation, and aging. Graph robustness includes resilience to noisy edges, adversarial perturbations, and distribution shift. Because graph convolutional networks aggregate information from neighbors, a small number of corrupted edges or nodes can influence many predictions. An adversary might inject fake sensors, alter relationships, or poison local updates. FPGA designs can include redundancy, error detection, or trusted execution, but these measures consume area and energy. The appropriate level of redundancy depends on the application. A smart factory may tolerate occasional classification errors if safety interlocks exist, while a medical graph may require stronger guarantees.

Sustainability is often framed as energy efficiency, but edge deployments have broader environmental implications. FPGAs can reduce operational energy by performing inference locally, yet their manufacturing carries embodied carbon. Reconfigurability can extend device lifetime by supporting new models, which may reduce replacement frequency. However, frequent remote updates and increasing model complexity can shorten useful life or encourage premature upgrades. A sustainable design should therefore consider total cost of ownership, repairability, and end-of-life recovery. It should also avoid rebound effects, where efficiency gains lead to more sensors, more data, and more energy consumption. Edge intelligence can support environmental applications, such as optimizing energy grids or reducing transportation emissions, but it can also intensify consumption if deployed without limits.

Fairness is especially complex for graph convolutional networks because graph structure encodes social, economic, and spatial relations. If a graph is built from biased data, neighborhood aggregation can amplify disadvantage. For example, a credit or employment graph may connect people through institutions that already exclude certain groups. An edge accelerator that processes such graphs efficiently does not correct the underlying bias; it may scale it. Fairness auditing methods for artificial intelligence emphasize the need to examine outcomes across groups and contexts [22]. In edge graph intelligence, auditing must also consider how partitioning, sampling, and local updates affect group outcomes. A model may perform well overall but poorly for a minority subgraph that is weakly connected. Hardware-level choices such as quantization can also disproportionately affect nodes with rare features. Fairness therefore cannot be treated as a post-processing step; it must inform graph construction, model design, accelerator precision, and deployment policy.

7. Case Illustrations and Cross-Domain Comparisons

Several domains illustrate the promises and tensions of FPGA-based graph convolutional acceleration. In smart cities, graph models can represent roads, transit routes, sensors, and incidents. An FPGA at an intersection or edge gateway could aggregate local traffic conditions and adjust signals with low latency. This reduces dependence on cloud connectivity and supports real-time response. However, the same infrastructure could be used for movement tracking or predictive policing if governance is weak. The technical efficiency of the accelerator does not determine whether the application is just; it only makes the application more feasible. Cross-domain comparison with cloud-based GPU acceleration shows that edge FPGAs may win on latency and privacy but lose on centralized model freshness and large-scale retraining. Hybrid designs that train centrally and infer locally are therefore likely.

In industrial systems, graph convolutional networks can model machines, pipelines, supply chains, and control dependencies. FPGA acceleration can support anomaly detection and predictive maintenance near equipment. Robustness requirements are high because false negatives may cause safety incidents and false positives may halt production. Reconfigurable logic can implement domain-specific feature extraction and protocol handling, which is difficult on general-purpose processors. Yet industrial deployments often have long lifecycles, so update governance and vendor support become critical. A proprietary accelerator that cannot be maintained after ten years may be less sustainable than a slower but well-documented alternative.

In healthcare, patient similarity graphs and clinical knowledge graphs can support diagnosis, triage, and care coordination. Edge FPGAs could process sensitive data within a hospital without sending it to a public cloud. This aligns with privacy principles and may reduce

latency for time-critical decisions. However, clinical validity, explainability, and liability require rigorous auditing. Fairness concerns are acute because graph connections may reflect historical inequities in access to care. A hardware-accelerated system may need to log decision pathways, support human review, and allow independent evaluation. These requirements may limit the use of opaque bitstreams or tightly coupled proprietary stacks. The comparison with general-purpose accelerators is instructive: GPUs may offer easier programmability and better support for rapidly changing models, while FPGAs may offer better energy and latency control for stable, well-characterized workloads.

8. Open Challenges and Research Directions

Several open challenges remain. First, programming productivity must improve. Developing FPGA accelerators for graph convolutional networks still requires expertise in graph algorithms, hardware design, compilers, and systems deployment. Higher-level abstractions are needed without sacrificing control over memory and parallelism. Second, dynamic graph support must mature. Most accelerators are evaluated on static graphs, but edge graphs change continuously. Runtime reconfiguration, incremental partitioning, and adaptive precision are promising but underexplored. Third, security and privacy require hardware-software co-design. Secure bitstreams, attestation, encrypted graph processing, and privacy-preserving aggregation must be integrated without excessive overhead. Fourth, benchmarking must become more holistic. Throughput and energy per inference are important, but so are latency tails, update cost, robustness, fairness, and maintainability. Fifth, interdisciplinary evaluation is necessary. Engineers, social scientists, policymakers, and domain experts must jointly assess deployments. A technically superior accelerator may fail if it violates privacy expectations or cannot be audited.

Future research should also explore heterogeneous edge clusters in which FPGAs, CPUs, GPUs, and specialized accelerators cooperate. Graph partitions could be assigned dynamically based on topology, privacy boundaries, and energy availability. Federated learning over graphs could reduce data movement, but it raises new questions about trust and aggregation. Sustainable design should consider modularity, repairability, and long-term support. Fairness-aware acceleration could allocate precision or latency budgets to protect vulnerable subgraphs. Policy frameworks should encourage transparency, security, and accountability without stifling beneficial innovation. The central insight is that FPGA-based graph acceleration for edge intelligence is not merely a hardware race; it is an infrastructure design practice with social consequences.

9. Conclusion

FPGA-based acceleration of graph convolutional networks offers a promising path toward edge intelligence because it can balance energy efficiency, latency control, and post-deployment adaptability. The technical case is strong for workloads that are sparse, irregular, and latency-sensitive, especially when cloud offloading is undesirable. However, the system-level case depends on more than raw performance. Memory hierarchy, quantization, graph partitioning, runtime orchestration, security, governance, robustness, sustainability, and fairness all shape whether an accelerator can be deployed responsibly and maintained over time. The analysis in this paper suggests that the most successful designs will be co-designed across algorithms, hardware, software, and institutions. They will support secure updates, transparent auditing, and adaptive precision, and they will be evaluated not only on benchmarks but also on their contributions to public value. Edge graph intelligence is therefore best understood as a socio-technical infrastructure in which FPGA acceleration is

one component among many. Future work should pursue integrated benchmarks, dynamic graph support, privacy-preserving execution, and fairness-aware deployment so that reconfigurable graph acceleration can serve both technical and societal goals.

References

1. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In 5th International Conference on Learning Representations (ICLR 2017). https://doi.org/10.48550/arXiv.1609.02907

2. Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems (pp. 1024–1034). Curran Associates.

3. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In International Conference on Learning Representations (ICLR 2018).

4. Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? In International Conference on Learning Representations (ICLR 2019).

5. 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

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. Abadal, S., Jain, A., Guirado, R., López-Alonso, J., & Alarcón, E. (2021). Computing graph neural networks: A survey from algorithms to accelerators. ACM Computing Surveys, 54(9), Article 191. https://doi.org/10.1145/3477141

9. Liang, S., Wang, Y., Liu, C., He, Z., Li, H., Xu, D., & Li, X. (2021). EnGN: A high-throughput and energy-efficient accelerator for large graph neural networks. IEEE Transactions on Computers, 70(9), 1511–1525. https://doi.org/10.1109/TC.2020.3014632

10. Yan, M., Deng, L., Hu, X., Liang, L., Feng, Y., Ma, X., Ye, H., & Xie, Y. (2020). HyGCN: A GCN accelerator with hybrid architecture. In 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA) (pp. 15–29). IEEE. https://doi.org/10.1109/HPCA47549.2020.00012

11. Garg, R., Qin, E., Muñoz-Matrínez, F., Guirado, R., Jain, A., Abadal, S., Alarcón, E., & Krishna, T. (2021). Understanding the design space of graph neural networks for machine learning accelerators. In 2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) (pp. 129–140). IEEE. https://doi.org/10.1109/ISPASS51385.2021.00028

12. Dave, S., Baghdadi, R., Shafiee, A., & Sadeghi, A. R. (2019). Hardware acceleration of graph neural networks. In 2019 56th ACM/IEEE Design Automation Conference (DAC) (pp. 1–6). IEEE. https://doi.org/10.1145/3316781.3323475

13. Zhang, C., Xu, Z., Lu, C., Liu, Y., & Sun, G. (2020). GNNAdvisor: An adaptive and efficient runtime system for GNN acceleration on GPUs. In 2020 USENIX Annual Technical Conference (ATC) (pp. 507–520). USENIX Association.

14. Xie, X., Peng, H., Hasan, A., Huang, S., Zhao, J., Fang, H., ... & Ding, C. (2023, October). Accel-gcn: High-performance gpu accelerator design for graph convolution networks. In 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD) (pp. 01-09). IEEE.

15. Umuroglu, Y., Fraser, N. J., Gambardella, G., Blott, M., Leong, P., Jahre, M., & Vissers, K. (2017). FINN: A framework for fast, scalable binarized neural network inference. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 65–74). ACM. https://doi.org/10.1145/3020078.3021744

16. Blott, M., Preußer, T. B., Fraser, N. J., Gambardella, G., O'Brien, K., Umuroglu, Y., Leeser, M., & Vissers, K. (2018). FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks. ACM Transactions on Reconfigurable Technology and Systems, 11(3), Article 16. https://doi.org/10.1145/3242897

17. Nurvitadhi, E., Sheffield, D., Sim, J., Mishra, A., Venkatesh, G., & Marr, D. (2016). Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC. In 2016 International Conference on Field-Programmable Technology (FPT) (pp. 77–84). IEEE. https://doi.org/10.1109/FPT.2016.7929192

18. Guo, K., Zeng, S., Yu, J., Wang, Y., & Yang, H. (2019). A survey of FPGA-based neural network inference accelerators. ACM Transactions on Reconfigurable Technology and Systems, 12(1), Article 2. https://doi.org/10.1145/3289185

19. Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, 105(12), 2295–2329. https://doi.org/10.1109/JPROC.2017.2761740

20. Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for VM-based cloudlets in mobile computing. IEEE Pervasive Computing, 8(4), 14–23. https://doi.org/10.1109/MPRV.2009.82

21. Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

22. Raji, I. D., & Buolamwini, J. (2019). Actionable auditing: Investigating the impact of publicly naming biased performance results of commercial AI products. In Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society (pp. 429–435). ACM. https://doi.org/10.1145/3306618.3314244