

Edge AI Acceleration for Real-Time Graph Neural Network Inference

Meral Fritsch

Department of Computer Science, University of Alabama at Birmingham, Birmingham, AL,
USA.

meral.work@uab.edu

Abstract

Real-time graph neural network inference at the edge has emerged as a critical systems challenge because graph-structured data increasingly mediates mobility, industrial monitoring, health sensing, communication networks, and urban infrastructure. Unlike conventional image or sequence inference, graph neural networks exhibit irregular memory access, dynamic neighborhood expansion, sparse feature aggregation, and strong dependencies between topology and computation. These properties complicate the deployment of low-latency inference on resource-constrained devices, where energy, thermal headroom, memory bandwidth, and intermittent connectivity are limited. This paper examines edge artificial intelligence acceleration for real-time graph neural network inference from a system-level perspective. It argues that meaningful acceleration requires co-design across hardware architecture, runtime orchestration, graph partitioning, model compression, deployment governance, and socio-technical policy. The discussion first clarifies the computational and infrastructural foundations of edge graph inference. It then analyzes architectural trade-offs among mobile graphics processors, neural processing units, field-programmable gate arrays, and application-specific accelerators. Subsequent sections address runtime infrastructure, distributed deployment across edge-cloud continua, robustness under noisy and adversarial conditions, fairness in graph-mediated decisions, sustainability, and policy implications. The paper concludes that real-time graph inference at the edge is not merely a kernel optimization problem but an institutional and infrastructural problem requiring adaptive acceleration, transparent evaluation, and accountable governance.

Keywords

edge artificial intelligence, graph neural networks, real-time inference, heterogeneous acceleration, distributed systems, fairness, sustainability, governance.

1. Introduction

Graph neural networks have become a dominant computational approach for learning from relational data because they can combine node features, edge structure, and neighborhood context within a unified representation. Their adoption has expanded from recommendation and social analysis into domains that demand immediate response, including intelligent transportation, collaborative robotics, industrial anomaly detection, wearable health monitoring, and distributed sensing. In these settings, inference often cannot be deferred to a distant cloud data center because latency, privacy, bandwidth, and availability constraints alter the value of the computation itself. Edge artificial intelligence therefore becomes less a matter of convenience and more a structural requirement for systems that must respond within physical, legal, or operational time windows [1], [2]. The central question is not whether

graph neural networks can run at the edge, but what forms of acceleration make such inference reliable, equitable, sustainable, and governable at scale.

The difficulty arises because graph neural networks are unusually hostile to conventional edge acceleration assumptions. Convolutional neural networks process dense tensors with predictable reuse, whereas graph neural networks operate over sparse and irregular structures whose access patterns depend on topology. Real-time inference compounds this irregularity because the system must satisfy latency targets under rapidly changing graph conditions. A transportation graph may shift as vehicles enter and leave intersections, an industrial sensor graph may change when devices fail, and a communication network graph may reconfigure under congestion. The edge system must therefore adapt not only its model execution but also its data movement, partitioning, scheduling, and energy management. These concerns connect directly to edge computing architectures that distribute computation near data sources while retaining coordination with centralized services [1], [2]. They also connect to broader graph learning research that has established the expressive power and practical limitations of neighborhood aggregation, attention, and message passing [3], [4].

A purely algorithmic view of graph neural network acceleration tends to underestimate the institutional and infrastructural dimensions of deployment. Edge devices are owned, maintained, regulated, and retired within social systems. Their acceleration capabilities are shaped by supply chains, standards, privacy regimes, spectrum policy, and public procurement. Real-time inference may be used to allocate mobility rights, prioritize maintenance, detect health risks, or control industrial machinery, which means that latency improvements can also amplify the consequences of biased or brittle predictions. This paper therefore treats edge acceleration as a socio-technical design problem. It examines how architectural choices, runtime systems, deployment models, and governance mechanisms interact. The argument is that real-time graph inference at the edge requires a layered strategy: efficient kernels and accelerators at the bottom, adaptive orchestration in the middle, and accountable policy and evaluation at the top.

2. Graph Neural Network Inference and Edge Constraints

Graph neural network inference is built around iterative neighborhood aggregation. A node representation is updated by combining information from adjacent nodes, and repeated layers expand the effective receptive field across the graph. Early formulations demonstrated that this message-passing pattern can support semi-supervised classification and inductive learning on large graphs [4], [5]. Attention-based variants further allow the model to weight neighbors differently, which improves flexibility but increases computational and memory pressure [6]. Survey work has documented the broad family of methods, including spectral approaches, spatial methods, recurrent graph models, and graph autoencoders, each with distinct inference profiles [3], [7]. For edge deployment, the important point is that these models are not merely smaller versions of dense neural networks. Their performance depends on graph structure, degree distribution, feature dimensionality, layer depth, and batching strategy.

The edge environment imposes constraints that are absent or less severe in data centers. Mobile and embedded devices often have limited memory bandwidth, constrained thermal envelopes, and finite battery capacity. They may rely on intermittent wireless links and heterogeneous accelerator support. A device may contain a graphics processor, a neural processing unit, a digital signal processor, or a programmable logic fabric, but no single accelerator is optimal for all graph operations. The irregularity of graph traversal can cause

poor cache utilization, load imbalance, and excessive synchronization. Real-time requirements intensify these issues because the system cannot simply wait for a large batch to accumulate. It must often process small batches, streaming updates, or even single-query subgraphs while maintaining predictable latency. Edge intelligence research has therefore emphasized the need to place computation close to data sources, but it has also shown that edge resources are not homogeneous or reliable [19].

A further constraint is the relationship between graph dynamics and model freshness. In many edge applications, the graph itself changes faster than the model can be retrained. Traffic, social interaction, and sensor networks evolve continuously, while model updates may occur hourly, daily, or weekly. Real-time inference must therefore tolerate stale parameters, incomplete neighborhoods, and missing features. The system may need to sample neighborhoods, cache frequently accessed embeddings, or approximate aggregation to meet deadlines [4], [8]. These approximations introduce trade-offs between accuracy, latency, energy, and fairness. A system that serves rapid but systematically degraded predictions for certain regions or device classes may satisfy average latency while violating equity or safety requirements. Consequently, edge graph inference must be evaluated not only by throughput and latency but also by distributional performance across contexts.

The computational foundations of graph neural networks also shape what acceleration can achieve. Survey work on computing graph neural networks distinguishes between algorithmic optimizations, such as sampling and partitioning, and hardware accelerators that exploit parallelism or sparsity [8]. Framework-level systems provide graph-centric abstractions that simplify implementation but may hide costly data movement [9]. Architectural studies have shown that graph neural network inference is often memory-bound rather than compute-bound, meaning that improvements in arithmetic units alone may not translate into proportional gains [11]. The edge context makes this especially salient because memory energy can dominate total energy consumption. Acceleration must therefore target locality, compression, scheduling, and data reuse, not just peak operations per second.

3. Architectural Trade-Offs in Edge Acceleration

The design space for edge graph acceleration spans general-purpose processors, mobile graphics processors, neural processing units, field-programmable gate arrays, and specialized application-specific integrated circuits. Each offers a different balance among flexibility, energy efficiency, programmability, and deployment risk. Mobile graphics processors are widely available and support parallel execution, but their memory hierarchies and scheduling policies are designed for dense graphics and tensor workloads. Neural processing units can deliver strong energy efficiency for regular operations, yet graph aggregation may not map cleanly onto their fixed data paths. Field-programmable gate arrays provide reconfigurable data movement and can exploit sparsity, but they require specialized expertise and may increase cost and verification burden. Application-specific accelerators can achieve high efficiency for known graph patterns, but they risk obsolescence when graph models or workloads change. These trade-offs are central to architectural research on graph neural network acceleration [8], [12], [13].

Recent work on graphics processor kernel design illustrates how much performance remains locked in data movement and workload balancing. Advanced kernel strategies can reduce redundant computation, improve memory coalescing, and adapt execution to graph sparsity, yet they must contend with irregular neighbor sets and varying degrees of parallelism [10]. Such kernel-level innovations are necessary but not sufficient for edge deployment. Edge

accelerators often have smaller caches, narrower memory buses, and stricter power limits than server graphics processors. A kernel that performs well on a data-center graphics processor may become energy-inefficient on a mobile device because it relies on abundant parallelism or large shared memory. Effective edge acceleration therefore requires rethinking kernel fusion, tile sizing, compression formats, and scheduling under thermal constraints. The goal is not simply to port a server kernel but to redesign the execution model for a constrained and dynamic device.

Hybrid architectures attempt to reconcile flexibility with efficiency by assigning different graph operations to different processing elements. For example, aggregation and feature transformation have different computational profiles, and partitioning them across specialized units can reduce idle time and memory pressure [12]. Runtime workload rebalancing is another important direction because graph degree distributions are often skewed. A small number of high-degree nodes can dominate computation while many low-degree nodes remain underutilized [13]. Adaptive runtime systems can detect imbalance and redistribute work across threads, devices, or partitions [14]. At the edge, such adaptivity must operate with limited observability and unpredictable connectivity. The accelerator cannot assume that a central scheduler has complete knowledge of the graph. It must infer local conditions, adjust execution, and degrade gracefully when resources become unavailable.

Memory hierarchy design is perhaps the most consequential architectural trade-off. Graph inference repeatedly gathers neighbor features, combines them, and writes updated representations. If features are stored in external memory, the cost of repeated access can overwhelm compute. On-chip caching, compression, and recomputation can reduce memory traffic, but they introduce complexity and may harm accuracy or generality. Quantization and low-precision arithmetic can improve energy efficiency, yet graph aggregation may be sensitive to numerical error, especially when many neighbors contribute to a single update. Architectural studies have shown that graph neural networks stress conventional cache and prefetching mechanisms because access patterns are data-dependent [11]. Edge accelerators must therefore combine flexible memory controllers, sparse data formats, and locality-aware scheduling. They must also support graceful fallback to cloud or nearby edge servers when local resources are insufficient.

Cross-domain comparisons reveal that graph acceleration differs from vision and language acceleration in important ways. Vision workloads benefit from spatial locality and regular convolution, while language workloads benefit from sequence locality and predictable tensor shapes. Graph workloads have topology-dependent locality that can change between queries. A recommendation graph, a traffic graph, and a molecular graph may share message-passing abstractions but differ radically in degree distribution and feature density. Edge acceleration therefore cannot rely on a single static optimization. It requires a family of execution strategies selected by graph properties, latency targets, and device capabilities. This suggests that future edge accelerators should expose richer runtime interfaces, allowing the system to choose among sampling, partitioning, precision, and caching policies at inference time. Such interfaces also create governance needs, because adaptive optimization can produce variable service quality across users and regions.

4. Runtime Infrastructure, Orchestration, and Deployment

Real-time edge graph inference depends on runtime infrastructure that spans devices, gateways, edge servers, and cloud services. The edge-cloud continuum is not a single tier but a hierarchy of computational and storage resources with different latency, bandwidth, cost,

and trust characteristics. A device may perform initial filtering, a gateway may aggregate local subgraphs, an edge server may run deeper model layers, and the cloud may provide retraining or global coordination. Orchestration must decide where each portion of the graph computation should execute while respecting privacy constraints and service-level objectives. This resembles broader distributed machine learning systems, where parameter servers and communication scheduling determine scalability [18]. However, graph inference adds topology dependence: the best partition is not merely a matter of data size but of graph cuts, neighborhood overlap, and communication frequency.

Graph partitioning and batching are central runtime concerns. If a graph is partitioned poorly, many edges cross device boundaries, causing communication overhead that can dominate latency. If partitions are too small, accelerators are underutilized. If they are too large, memory and energy constraints are violated. Frameworks such as graph-centric libraries and geometric learning toolkits provide abstractions for loading, sampling, and executing graph operations [9], [15]. Benchmark suites such as the Open Graph Benchmark provide standardized datasets and evaluation protocols that help compare systems across scales and domains [16]. Yet benchmark performance does not automatically translate into edge performance, because edge workloads often involve streaming updates, heterogeneous devices, and real-time deadlines. Runtime systems must therefore support dynamic repartitioning, caching, and speculative execution, with careful attention to tail latency [17].

Tail latency is especially important in real-time graph inference. A system may have low average latency but still fail if a small fraction of queries exceed the deadline. Graph irregularity makes tail latency likely because high-degree nodes or dense subgraphs can create transient bottlenecks. Techniques from large-scale online services, such as hedged requests, replication, and priority scheduling, can help, but they must be adapted to edge constraints [17]. Replication consumes scarce memory and energy, while hedging may increase network traffic. A promising approach is to combine predictive scheduling with approximate computation: when a query is likely to miss its deadline, the system can return a lower-fidelity result from a cached or sampled subgraph. This requires explicit accuracy-latency contracts and monitoring to ensure that approximations are not systematically applied to vulnerable populations or safety-critical functions.

Deployment also involves lifecycle management. Edge devices may be updated infrequently, operate in uncontrolled environments, and lack skilled administrators. Model and accelerator updates must be validated for compatibility, security, and rollback. Federated learning and decentralized training can reduce the need to centralize sensitive graph data, but they introduce new risks such as poisoning, model inversion, and uneven participation [20]. Runtime infrastructure must therefore include provenance tracking, secure boot, encrypted communication, and policy enforcement. It must also support observability under privacy constraints, meaning that operators need aggregate performance metrics without accessing raw user data. These requirements connect technical orchestration to institutional trust. Without transparent monitoring and accountable update procedures, edge graph inference may face resistance from users, regulators, and operators even when latency improves.

5. Robustness, Fairness, and Governance

Robustness is a first-order requirement for real-time graph inference because edge environments are noisy and adversarial. Sensor failures, missing links, stale features, and malicious perturbations can alter the graph structure and degrade predictions. Graph neural networks are vulnerable to structural attacks, including edge insertion, edge deletion, and

feature manipulation, because message passing propagates corrupted information across neighborhoods [3], [7]. Real-time constraints limit the time available for expensive defense mechanisms, so edge systems must rely on lightweight anomaly detection, uncertainty estimation, and graceful degradation. A robust system should detect when the input graph deviates from expected distributions and adjust its confidence or fallback strategy. It should also distinguish between benign distribution shift and adversarial manipulation, although this distinction is difficult under partial observability.

Fairness in graph-mediated inference raises distinctive challenges. Graph neural networks can amplify structural inequalities because predictions depend on neighborhood composition. If certain communities are underrepresented in the graph, have fewer connections, or experience noisier data, their node representations may be systematically less accurate. Real-time acceleration can worsen fairness if it allocates more computation to high-value or high-connectivity regions while approximating others. For example, a traffic inference system might prioritize dense urban areas with better sensor coverage and provide delayed or degraded service in peripheral areas. A health monitoring system might perform better for users with rich interaction histories than for isolated users. Fairness therefore cannot be treated as a post hoc audit of model outputs. It must be designed into scheduling, partitioning, and approximation policies. Governance mechanisms should require disaggregated performance reporting across regions, device classes, and demographic proxies where legally permissible.

Governance also concerns privacy and data sovereignty. Edge inference is often justified as a privacy-preserving alternative to cloud processing, but graph data can reveal sensitive relationships even when raw features are not transmitted. Federated learning mitigates some risks by keeping data local, yet gradients or model updates can still leak information [20]. Real-time systems may also generate fine-grained mobility, health, or behavioral traces that are valuable for surveillance or commercial profiling. Policymakers must therefore address purpose limitation, retention, access control, and redress. Technical mechanisms such as differential privacy, secure aggregation, and trusted execution environments can help, but they impose performance costs that interact with real-time acceleration. The design challenge is to allocate privacy budgets and security resources without making latency guarantees meaningless. This requires explicit trade-off governance rather than assuming that edge deployment is inherently ethical.

The governance of edge artificial intelligence must also consider accountability across distributed components. When a graph inference failure occurs, responsibility may be spread across device manufacturers, network operators, model developers, data providers, and public agencies. Real-time systems can make this problem worse because decisions are automated and difficult to reconstruct after the fact. Logging every message and update may be infeasible due to privacy and storage constraints, while logging too little undermines auditability. A balanced approach uses selective provenance, cryptographic attestation, and aggregate performance monitoring. Regulators may need to require impact assessments for high-risk graph applications, especially those involving mobility, employment, healthcare, or public safety. These assessments should consider not only average accuracy but also worst-case latency, failure modes, and distributional outcomes. In this sense, acceleration and accountability are not separate concerns; they are linked through the architecture of the system.

6. Sustainability, Policy, and Cross-Domain Implications

Sustainability is an increasingly important dimension of edge acceleration. Moving computation to the edge can reduce network energy and data center load, but it also distributes energy consumption across many devices with shorter lifespans and less efficient cooling. The net carbon benefit depends on device manufacturing, utilization, and replacement cycles. Real-time graph inference may require continuous sensing and communication, which can dominate energy use even when the accelerator is efficient. Architectural choices such as low-precision arithmetic, sparsity exploitation, and adaptive duty cycling can reduce energy, but they must be evaluated over the entire lifecycle. A system that achieves low latency by using larger batteries or more frequent hardware replacement may be environmentally worse than a slightly slower cloud-assisted design. Policy frameworks should therefore encourage lifecycle assessment and transparent energy reporting for edge artificial intelligence deployments [19].

Policy also shapes the infrastructure of edge graph inference through spectrum allocation, interoperability standards, and public procurement. Wireless bandwidth is a shared resource, and real-time graph applications may compete with other services. Standards for accelerator interfaces, model formats, and privacy-preserving computation can reduce fragmentation and improve portability, but they can also entrench dominant vendors. Public agencies that deploy edge artificial intelligence for traffic, welfare, or policing must consider procurement rules that require auditability, competition, and community participation. Data sovereignty rules may prohibit cross-border transmission of certain graph data, encouraging local edge processing but also creating uneven capabilities across regions. These policy choices influence which communities receive low-latency services and which become dependent on distant or unreliable infrastructure.

Cross-domain comparisons show both the promise and the risk of real-time graph inference. In transportation, accelerated graph models can coordinate traffic signals, predict congestion, and support autonomous mobility, but they may also enable fine-grained movement tracking. In healthcare, wearable and clinical graphs can detect deterioration or adverse events, yet errors can have severe consequences and privacy risks are acute. In industrial systems, graph-based anomaly detection can prevent equipment failure, but adversarial manipulation or sensor drift may cause unsafe shutdowns or missed hazards. In agriculture and environmental monitoring, edge graph inference can support distributed sensing under intermittent connectivity, but resource constraints may limit model complexity. Across these domains, the common pattern is that acceleration changes the feasible governance boundary. Faster inference enables more automated action, which raises the stakes of robustness, fairness, and accountability.

Forward-looking perspectives suggest several priorities. First, edge graph accelerators should be adaptive rather than fixed, with runtime selection among precision, sampling, partitioning, and caching strategies. Second, benchmark suites should include real-time, streaming, and edge-specific metrics, not only static accuracy and throughput [16]. Third, runtime systems should expose policy hooks that allow operators to enforce fairness, privacy, and energy constraints. Fourth, governance should require continuous monitoring and redress mechanisms for deployed systems. Fifth, research should examine the interaction between acceleration and social equity, especially when latency becomes a condition for access to services. These priorities connect technical design to the broader question of how computational speed is distributed and who benefits from it [19], [20].

7. Conclusion

Edge artificial intelligence acceleration for real-time graph neural network inference is a systems problem that spans kernels, architectures, runtimes, infrastructure, and governance. Graph neural networks offer powerful relational inference, but their irregular memory access, dynamic topology, and neighborhood-dependent computation make them difficult to accelerate on resource-constrained devices. Effective solutions must therefore combine specialized hardware, adaptive runtime orchestration, graph-aware partitioning, compression, and energy management. They must also recognize that latency guarantees are not merely technical achievements; they determine which applications can operate safely, which users receive timely service, and which institutions can automate decisions. The analysis in this paper suggests that no single accelerator or optimization can resolve the full design space. Instead, progress depends on layered co-design that connects device-level efficiency to system-level resilience and societal accountability.

The most promising direction is adaptive edge acceleration that treats latency, accuracy, energy, privacy, and fairness as jointly managed resources. Such systems would select execution strategies based on graph properties, device conditions, and policy constraints. They would monitor tail latency and distributional performance, not only average throughput. They would support federated and decentralized coordination while preserving auditability and redress. They would also be evaluated through lifecycle sustainability metrics and cross-domain benchmarks. Real-time graph inference at the edge is likely to become more common as sensing and connectivity expand. Whether it becomes trustworthy will depend on whether acceleration research and policy development proceed together rather than in isolation.

References

1. Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1), 30–39. <https://doi.org/10.1109/MC.2017.9>
2. 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>
3. 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>
4. Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 1024–1034.
5. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*.
6. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. *International Conference on Learning Representations*.
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), 1–38. <https://doi.org/10.1145/3477141>

9. Wang, M., Yu, L., Zheng, D., Gan, Q., Gai, Y., Ye, Z., Li, M., 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. <https://arxiv.org/abs/1909.01315>
10. 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).
11. Zhang, Z., Leng, J., Ma, X., Miao, Y., Li, C., & Guo, M. (2020). Architectural implications of graph neural networks. *IEEE Computer Architecture Letters*, 19(1), 59–62. <https://doi.org/10.1109/LCA.2020.2988991>
12. Yan, M., Deng, L., Hu, X., Liang, L., Feng, Y., Ma, X., ... & Xie, Y. (2020). HyGCN: A GCN accelerator with hybrid architecture. In *2020 IEEE International Symposium on High Performance Computer Architecture* (pp. 15–29). IEEE. <https://doi.org/10.1109/HPCA47549.2020.00012>
13. Geng, T., Li, A., Shi, R., Wu, C., Wang, T., Li, Y., ... & Herbordt, M. C. (2020). AWB-GCN: A graph convolutional network accelerator with runtime workload rebalancing. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture* (pp. 922–936). IEEE. <https://doi.org/10.1109/MICRO50266.2020.00079>
14. Wang, Y., Feng, B., Li, G., Li, S., Deng, L., Xie, Y., & Ding, Y. (2021). GNNAdvisor: An adaptive and efficient runtime system for GNN acceleration on GPUs. In *15th USENIX Symposium on Operating Systems Design and Implementation* (pp. 515–531). USENIX Association.
15. Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
16. 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. *Advances in Neural Information Processing Systems*, 33, 22118–22133.
17. Dean, J., & Barroso, L. A. (2013). The tail at scale. *Communications of the ACM*, 56(2), 74–80. <https://doi.org/10.1145/2408776.2408794>
18. Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., & Su, B.-Y. (2014). Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation* (pp. 583–598). USENIX Association.
19. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., & Zhang, J. (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8), 1738–1762. <https://doi.org/10.1109/JPROC.2019.2918951>
20. Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., & Srivastava, G. (2021). A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115, 619–640. <https://doi.org/10.1016/j.future.2020.10.007>