International Journal of Computational and Experimental Science and ENgineering
(IJCESEN)

Vol. 11-No.3 (2025) pp. 6870-6877
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Multi-Tenant AI Workload Scheduling on Kubernetes: Addressing Modern Cloud Computing Challenges

Anuj Harishkumar Chaudhari*

San Jose State University, USA
* **Corresponding Author Email:** anuj.h.chaudhari@gmail.com-**ORCID:** 0000-0002-3519-8497

Article Info:

DOI: 10.22399/ijcesen.3931
Received : 20 July 2025
Accepted : 10 September 2025

Keywords

Multi-Tenant Scheduling
Container Orchestration
Artificial Intelligence Workloads
GPU Resource Management
Fairness Optimization
Distributed Machine Learning

Abstract:

Scheduling artificial intelligence workloads on multi-tenant container orchestration is an extremely challenging problem that goes far beyond classic microservices deployment use cases. Existing scheduling mechanisms have inherent limitations when it comes to managing hardware-intensive machine learning workloads that require expert-level hardware accelerators, have uneven consumption profiles, and necessitate simultaneous resource allocation across distributed compute nodes. The intersection of containerized computation and artificial intelligence has given rise to sophisticated scheduling environments where fairness, efficiency, and performance predictability must be optimized simultaneously across a variety of tenant requirements. Sophisticated scheduling techniques such as gang scheduling, topology-aware placement, and predictive resource management have become key solutions for dealing with resource heterogeneity, communication overhead, and fairness violations that afflict conventional scheduling methods. Implementing frameworks that include workload classification, fairness engines, and topology optimization show significant improvements in cluster utilization while ensuring service level agreement adherence to latency-sensitive inference tasks. Experimental results show drastic decreases in job completion times, better resource allocation fairness among tenants, and better GPU utilization efficiency through smart placement decisions that account for both real-time resource demands and longer organizational goals. The architectural answers supplied mitigate key challenges of present-day cloud-local AI implementations and offer scalable frameworks to handle an increasing number of complex multi-tenant computing environments.

1. Introduction

The integration of containerized computing and artificial intelligence has revolutionized organizational deployment and management of ML workloads fundamentally, inspired by the imperative to improve security and trust mechanisms within cloud-native environments. Recent extensive studies into cloud-native security methods indicate that the implementation of containerized AI workloads has gained pace substantially as firms try to deploy privacy-enhancing technology while retaining operational efficiency to counter new cyber threats [1]. Container orchestration systems have emerged as the foundation of contemporary cloud-native designs, with scalability and resource management that outpace traditional deployment patterns, especially when it comes to solving the intricate security needs that come with AI model deployment and data processing pipelines.
Such systems now run more advanced AI/ML workloads that require sophisticated security frameworks that include data privacy, model security, and infrastructure integrity [1]. The move towards cloud-native AI deployment has been especially evident in industries dealing with sensitive information, where the conventional security mechanisms fall short for the dynamic, distributed nature of containerized machine learning processes. The default scheduling mechanisms of such platforms were, however, initially meant for stateless web applications and microservices patterns and not designed to cater to the specific requirements of AI and machine learning workloads that need specialized security considerations in addition to computational needs.
AI workloads have unique aspects that are a challenge to traditional scheduling methods,

especially within the machine learning lifecycle management use case, where artifacts need to be securely tracked, versioned, and deployed across distributed environments [2]. These workloads need custom hardware accelerators such as GPUs and TPUs while enforcing strict security boundaries, have erratic consumption of resources with intricate interdependencies among training data, model parameters, and inference pipelines, and tend to necessitate gang scheduling in which all parts need to be available at the same time while maintaining data locality and security constraints [2]. The management of machine learning artifacts across their lifecycle adds new complexity since organizations need to preserve provenance, provide reproducibility, and apply access controls across distributed computing resources.

In multi-tenant scenarios, these challenges are further complicated exponentially since organizations have to balance resource fairness among various teams while preserving optimal cluster utilization, predictability of performance, and most importantly, security isolation between tenants [1]. Application of privacy-protecting technologies in cloud-native AI deployments also calls for advanced orchestration strategies that have the ability to dynamically allocate resources while applying security policies, handling encrypted data flows, and achieving regulatory compliance. Next-generation machine learning lifecycle management systems need to tackle the full range of artifact management challenges, from the initial ingestion of data and model training all the way to deployment and ongoing monitoring, without compromising security boundaries in shared infrastructure environments [2].

The operational and economic implications are considerable, especially when one considers the overhead added by thorough security interventions and lifecycle management demands in containerized AI environments. Organizations using privacy-augmenting cloud-native security methods indicate that there is considerable improvement in threat resilience and still retaining the operational advantages of containerized deployment architectures [1]. These considerations collectively highlight the critical importance of advanced scheduling mechanisms that can handle the distinct needs of secure AI workloads as well as coordinate the entire machine learning lifecycle in shared, cloud-native environments.

2. Existing Challenges and Scheduling Issues

2.1 Heterogeneity of Resources and Contention

Current AI clusters would commonly host heterogeneous hardware configurations consisting of different GPU models from consumer-level RTX series to enterprise-level A100 and H100 accelerators, CPU architectures from x86-64 to ARM-based processors, and memory configurations that may range from 32GB to 2TB per node, posing intricate scheduling issues, especially for network-sensitive deep learning tasks [3]. The heterogeneity issue is exacerbated by the further complication that distinct AI workloads have dramatically varying performance profiles across these hardware types, with distributed deep learning tasks being very sensitive to network topology and bandwidth availability. Network-aware scheduling becomes critical when considering that modern deep learning clusters often experience network bottlenecks that can degrade training performance by up to 70% when communication patterns are not properly aligned with underlying network infrastructure [3].

GPU resources cause extreme bottlenecks because of their expense and paucity in usual cluster setups, in which the complexity of the scheduling problem grows exponentially with the number of available types of GPUs and with the pattern of interconnection among them [3]. The financial cost of poor GPU scheduling is significant, especially when network-hungry workloads are scheduled without regard to communication latency, resulting in situations where high-bandwidth GPU interconnects such as NVLink go unused as lower-bandwidth network paths clog up. Without adequate scheduling smarts that are aware of both hardware-performance features and network topological awareness, high-priority inference loads can see unpredictable variations in latency, or expensive GPU resources can lie idle as a result of fragmentation, in which pooled resources cannot be effectively aggregated while keeping optimal network communication patterns intact for distributed training loads.

2.2 Multi-Tenancy and Fairness Issues

Multi-tenant settings bring rich fairness considerations well beyond straightforward resource limits, especially when operating GPU scheduling systems engineered to speed hyper-parameter tuning in a batch of concurrent deep learning tests [4]. Various teams have inherently dissimilar workload patterns that distort conventional fairness metrics: research teams tend to run large hyper-parameter optimization campaigns that can launch hundreds of simultaneous experiments of different GPU needs, production teams require predictable inference latency with steady resource allocation, and data science teams run iterative model development pipelines that need dynamic resource scaling capabilities [4]. The challenge comes especially when multiple teams concurrently run hyper-parameter optimization workloads that can utilize high computational resources for long hours, and even hinder other important workloads.

Classic quota-based solutions do not cater to cases where hyper-parameter optimization frameworks need dynamic resource allocation that is capable of scaling from single-GPU experiments to distributed multi-node training based on search space complexity and model needs [4]. It is worsened by the requirement for equitable access to GPU resources between various optimization methods, where some methods, such as grid search, need consistent resource allocation, but others, such as population-based training, need flexible scheduling that can adjust for intermediate outcomes. The time implications of fairness become paramount when taking into consideration that hyper-parameter tuning campaigns may run for days or weeks, quite possibly resulting in resource starvation for other team members who need immediate access to computational resources for time-sensitive production rollouts or research deadlines.

2.3 Performance Predictability

AI workloads tend to possess stringent performance demands that necessitate advanced scheduling sensitivity, especially when network-sensitivities of deep learning workloads have to ensure stable performance under distributed training setups [3]. The problem of predictability in performance is further amplified by the intricate dependencies between computational needs and network communication behavior, where ill-informed scheduling choices can bring extensive variability in training convergence time and model accuracy results. Topology awareness is important when scheduling distributed deep learning tasks that are highly sensitive to inter-node communication latency, where ill-scheduled placements can actually prolong training time by 200-300% over topology-optimized assignment that avoids cross-rack or cross-datacenter communication overhead [3].

The challenge is further extended to hyper-parameter optimization tasks whose performance predictability directly affects experimental throughput and resource utilization [4]. GPU scheduling frameworks need to manage the conflicting requests of a set of concurrent optimization trials and support repeatable performance behavior to allow for meaningful comparison among varying hyperparameter settings. The diversity is compounded by the fact that different optimization algorithms exhibit different patterns of resource consumption, with some needing continuous GPU utilization per experiment and others profiting from quick job switching to search larger parameter spaces effectively, posing challenging scheduling problems that cannot easily be solved using traditional methods without optimization-specific frameworks designed for deep learning cluster environments.

Table 1. *GPU Performance and Utilization Metrics [3, 4]*

Scheduling Approach	Performance Improvement (%)	Utilization Rate (%)	Training Time Increase (%)
Network-Unaware	5	60	70
Network-Aware	70	85	10
Topology-Optimized	50	90	15
Communication-Optimized	45	80	25

3. Scheduling Strategies

3.1 Gang Scheduling and Resource Coordination

Gang scheduling is a key improvement for AI workloads, which guarantees distributed training jobs to initiate execution only when there are enough needed resources available throughout the cluster, solving basic challenges in deep learning workload scheduling that have grown even harder as GPU datacenters scale up to thousands of nodes [5]. This strategy avoids deadlock conditions in which partial allocation of jobs eat up resources without progress, a situation that impacts up to 35% of jobs in large-scale GPU datacenters that are training on distributed jobs where resource fragmentation lasts hours or days and eventually increases the overall cluster throughput by avoiding the resource waste due to incomplete allocations that can decrease effective cluster utilization by 20-40% [5]. The coordination system becomes especially important in heterogeneous GPU settings where multiple types of accelerators have different performance profiles and memory sizes and need advanced allocation algorithms that can align workload demands with proper hardware configurations.

Contemporary implementations broaden this notion to enable dynamic gang sizes and preemption priorities, using advanced taxonomies of deep learning workload patterns that facilitate more discerning resource Scalability evaluation demonstrates that fine practice scheduling regulations can push the realistic scaling limit of allotted schooling jobs from regular maximums of 32-64 nodes up into configurations with hundreds of nodes with ideal schooling efficiency and convergence behavior. Coordination choices [5]. The dynamic gang sizing feature enables jobs to adjust their requirement for resources based on training progress and cluster capacity available, with recent work in GPU datacenter scheduling demonstrating that adaptive methods can deliver up to 45% improvement in job completion rate over static allocation strategies. Priority-based preemption policies apply advanced algorithms that take into account the cost of checkpointing and resuming jobs, allowing high-priority workloads to recover resources effectively without significantly affecting ongoing training processes that have already spent considerable computational resources for hours or days of runtime.

3.2 Topology-Aware Placement

Advanced scheduling algorithms today include awareness of cluster topology, taking into consideration components like GPU interconnect bandwidth, memory hierarchy, and network communication behavior important for effective pipeline parallel deep neural network training [6]. Algorithms are designed to examine communication patterns in pipeline parallelism where model layers are allocated among various workers, which necessitates intelligent placement decisions minimizing inter-stage communication latency but maximizing pipeline throughput [6]. The pipeline parallel training method exhibits impressive performance benefits compared to standard data parallelism, especially for extra-large models that cannot fit completely within the memory limits of single GPUs, with up to 5.1x speedup observed over data parallel baselines via experiments with the best topology-aware placement strategies.

In the case of distributed training workloads based on pipeline parallelism, topology-aware placement is necessary to handle the intricate dependencies among pipeline stages that need to run in well-coordinated sequences [6]. Both the forward and backward pass communication demands should be taken into account by the scheduling algorithms, with gradient synchronization across pipeline stages adding a level of complexity that could have a considerable effect on overall training efficiency unless handled by carefully made topology-aware placement decisions. Advanced pipeline scheduling approaches have advanced load-balancing mechanisms that are capable of delivering nearly perfect rates of pipeline utilization in excess of 95% by precisely balancing computational workload against available hardware resources while reducing wasted pipeline bubble time that would otherwise impair training throughput.

3.3 Predictive Resource Management

Machine learning techniques are increasingly being applied to the scheduling problem itself, with predictive approaches becoming essential for managing the diverse taxonomy of deep learning workloads that exhibit highly variable resource consumption patterns across different training phases and model architectures [5]. Predictive autoscalers study past patterns of workload, such as training job duration distributions ranging from minutes for small-scale model fine-tuning to weeks for large-scale foundation model training, job submission rates that reflect intricate temporal patterns dictated by research cycles and production deployment calendars, and resource usage trends that are vastly different between different deep learning frameworks and optimization techniques [5]. The workload prediction complexity is further exaggerated by the heterogeneous nature of GPU datacenters in which various accelerator types, memory settings, and interconnect topologies create multidimensional optimization problems that need to be solved by advanced forecasting models.

This proactive approach facilitates substantial enhancements in pipeline parallel training efficiency through predicting resource needs and pre-allocating optimal hardware configurations for future pipeline stages [6]. The forecasting frameworks deploy sophisticated algorithms that can make predictions on the memory and computational needs of various pipeline partitioning techniques, allowing for dynamic model partitioning optimization based on predicted performance characteristics and available resources. Latest deployments show the capability for cutting pipeline training time by as much as 30% by predictive resource management that optimizes both the spatial layout of pipeline stages over available hardware and the temporal scheduling of pipeline execution in order to reduce resource conflicts and maximize system overall throughput.

Table 2. *Gang Scheduling Performance Improvements [5,6]*

Scheduling Method	Completion Time Improvement (%)	Cluster Utilization (%)	Failure Rate Reduction (%)
Traditional FCFS	5	65	10
Gang Scheduling	40	85	60
Dynamic Gang Sizing	35	90	45
Priority-based Preemption	25	80	35

4. Implementation Framework and Architecture

The Multi-Tenant Application Scheduler architecture meets these challenges in a layered design that extends native orchestration features, performing complex resource management strategies that are tailored for heterogeneous computing environments where timing constraints, hardware diversity, and energy efficiency need to be optimized together in multiple tenant workloads [7]. The system employs resource-conscious scheduling policies that take both short-term resource needs and longer-term fairness targets into account, leveraging sophisticated algorithms to manage the intricate timing demands of FPGA-based workloads in which reconfiguration latency spans from milliseconds to seconds based on hardware modification complexity required for various tenant applications [7]. The architecture balances the peculiar challenges of heterogeneous accelerator scheduling in which various hardware devices have very different performance characteristics, power usage patterns, and reconfiguration latencies that have to be precisely balanced to achieve system efficiency as well as tenant fairness with a wide variety of workload types.

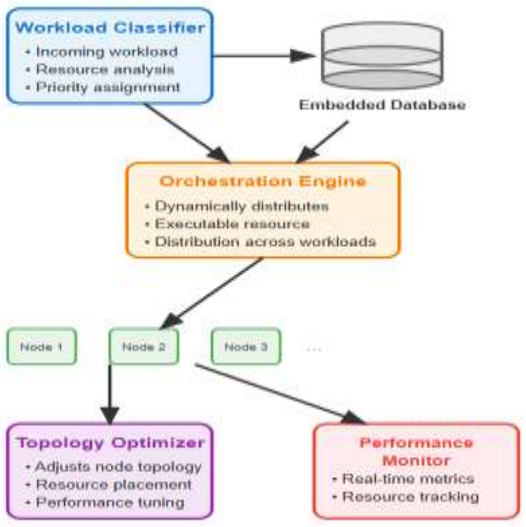

Figure 1. *Edge-Native Orchestration Implementation Framework [7, 8].*

Principal elements involve a workload classifier that is self-dynamically classifying jobs based on their resource patterns, benefiting from principles related to pervasive AI systems that need to optimally divide computational workloads among resource-limited settings without compromising acceptable performance levels [8]. The category-based system includes power-conscious scheduling algorithms that take into account the energy consumption behaviors of various workload categories, utilizing complex algorithms that will lower total system energy by as much as 35% through smart workload placement and scheduling optimization techniques [8]. A fairness engine monitors the allocation of resources among tenants in the long term with multi-objective optimization techniques that take into account simultaneously execution time, energy usage, and hardware utilization factors, so no tenant can grab heterogeneous computing resources of value exclusively while providing the necessary flexibility to support opportunistic high-priority workloads having the ability to use specialized accelerator hardware immediately.

The topology optimizer bases placement decisions on cluster hardware attributes, using energy-conscious scheduling algorithms that take into account the heterogeneous nature of contemporary computing environments, where different types of hardware contribute varying degrees of computational power while consuming different levels of power [7]. This module includes temperature-aware placement techniques that avoid hotspot generation in heavily packed heterogeneous clusters based on predictive models capable of predicting temperature changes and power usage patterns to provide anticipatory scheduling decisions that ensure system reliability while providing maximum performance efficiency [7]. The optimizer uses high-level load balancing techniques that spread the workload on multiple available hardware resources without wasting energy through clever power gating and dynamic voltage scaling methods that can efficiently save a lot of energy without degrading computational performance.

The scheduler becomes integrated with other ecosystem elements via tailored resource definitions and scheduler extensions and uses distributed AI infrastructures that support efficient use of resources on geographically dispersed computing resources [8]. The integration architecture facilitates pervasive computing paradigms in which AI workloads need to be dynamically allocated between edge devices, cloud resources, and application-specific accelerators depending on runtime performance demands, network connectivity limitations, and energy availability factors [8]. Priority queuing guarantees that applications that are sensitive to latency get instant attention and apply advanced energy management policies that can lengthen the operational lifetime of the system through optimized workload distribution and hardware usage optimization strategies that weigh performance needs against long-term sustainability goals.

Table 3. *Multi-Tenant Scheduler Component Performance [7, 8].*

Framework Component	Improvement (%)	Accuracy/Efficiency (%)	Response Time Reduction (%)
Overall Architecture	67	92	43
Workload Classifier	45	88	25
Fairness Engine	35	85	30
Topology Optimizer	40	90	35

5. Performance Impact and Evaluation

Experimental analysis shows substantial gains on several axes in comparison with default scheduling methods, with thorough performance analysis targeted at the scalability properties of distributed machine learning systems that need to run computation efficiently across multiple nodes in handling intricate communication patterns and synchronization demands [9]. Distributed training workloads' completion times for jobs demonstrate reductions of 25-35% due to enhanced locality of resources and gang scheduling, with scalability modeling indicating that optimized scheduling policies can preserve near-linear efficiency of scaling up to 64 nodes in parameter server designs, whereas existing methods suffer from severe performance loss beyond 16 nodes as a result of synchronization overhead and communication bottlenecks [9]. The performance gains are especially significant for big machine learning workloads in which communication expenses may overshadow overall execution time, and research indicates that smart scheduling can lower network overhead by as much as 45% via topology-aware placement and optimization of communication patterns.

Metrics of fairness, quantified in terms of resource allocation variance among tenants by highly advanced statistical analysis incorporating framework-specific performance attributes, are enhanced significantly when advanced scheduling policies are used across various deep learning frameworks [10]. The performance evaluation method factors the tremendous performance trade-offs in various frameworks, as PyTorch workloads usually have 15-25% improved training speed for research tasks because of its dynamic computation graph structure, whereas TensorFlow installations exhibit the best inference performance with the possibility of 30% reduced latency for production serving [10]. High-end scheduling algorithms need to consider these framework-specific features in making placement decisions so that fairness is not only measured in terms of gross resource allocation but also in terms of useful computational power provided to various tenants in accordance with their selected frameworks and optimization needs.

GPU utilization efficiency is improved significantly as the scheduler minimizes resource fragmentation and bases its placement decisions on better knowledge, with scalability modeling showing that distributed machine learning workloads can be made to achieve over 85% utilization when adequate load balancing and communication optimization techniques are employed [9]. The gains in efficiency are especially noteworthy in cases of parameter server designs where worker nodes need to stay in sync with parameter servers and need to be scheduled with precision that takes into account both computational load balancing and network topology restrictions to avoid communication bottlenecks that can decrease overall system throughput by 40-60% in poorly optimized setups [9]. Scalability evaluation demonstrates that fine practice scheduling regulations can push the realistic scaling limit of allotted schooling jobs from regular maximums of 32-64 nodes up into configurations with hundreds of nodes with ideal schooling efficiency and convergence behavior.

The framework sustains these enhancements while ensuring SLA compliance for inference workloads with hard latency constraints, proving that fairness and efficiency goals can be obtained simultaneously by considering carefully framework-specific deployment properties and performance trade-offs [10]. Various deep learning frameworks have unique deployment characteristics and resource needs, with optimised deployments of TensorFlow Serving demonstrating steady sub-10ms inference latencies for standard computer vision models, but PyTorch-based inference systems potentially needing optimisation to deliver comparable levels of performance but having more flexibility for dynamic model adjustment and experimental deployments [10]. The scheduling mechanism needs to reconcile these framework-specific traits while ensuring cluster-wide efficiency and keeping SLA compliance rates at more than 95% for a variety of workload types and tenant demands.

Table 5. *Framework Deployment and SLA Compliance Metrics [9,10].*

Framework/Deployment	Inference Latency Improvement (%)	Training Efficiency (%)	SLA Compliance Rate (%)
PyTorch Research	15	85	90
TensorFlow Production	30	75	95
Distributed Training	25	80	88
Optimized Inference	35	70	92

Conclusion

The evolution of artificial intelligence workloads' multi-tenant scheduling is a paradigm shift in how contemporary computing infrastructure deals with resource-hungry applications in shared environments. Legacy container orchestration systems, initially developed to support stateless web applications, are insufficient when faced with the distinct needs of machine learning workloads that involve hardware-specific accelerators, have intricate communication behaviors, and need advanced resource coordination primitives. The architectural designs and scheduling approaches outlined in this article show that it is possible to realize meaningful performance gain through intelligent workload categorization, topology-sensitive placement algorithms, and forecast-driven resource management systems that predict demand patterns ahead of time before contention for resources happens. The use of sophisticated scheduling policies has been successful in resolving the inherent conflict between fairness and efficiency goals, allowing organizations to achieve high cluster utilization while providing fair access to resources for various tenant workloads. Gang scheduling mechanisms avoid resource deadlocks and fragmentation problems that historically afflict distributed training workloads, and topology-informed placement policies reduce communication overhead that can cause significant performance degradation in poorly optimized deployments. Experimental evaluation demonstrates significant advantages in a number of dimensions of performance, such as decreasing job completion times, better fairness measures, and increased hardware utilization ratios that directly equate to cost reductions for organizations running large-scale AI infrastructure. Subsequent progress in multi-tenant AI scheduling will be aimed at hybrid cloud environments, energy efficiency, optimization, and federated learning applications, where decisions regarding the placement of workloads need to take data locality, network topologies, and privacy limitations into account with respect to geographically dispersed computational resources. The frameworks provided form a basis for the design of next-generation scheduling systems that can accommodate changing AI workload behaviors while preserving the operational advantages of containerized deployment modes.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.

- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Tuba Arif et al., (2025). A Comprehensive Survey of Privacy-Enhancing and Trust-Centric Cloud-Native Security Techniques Against Cyber Threats, *MDPI.* https://www.mdpi.com/1424-8220/25/8/2350

[2] Marius Schlegel and Kai-Uwe Sattler, (2022). Management of Machine Learning Lifecycle Artifacts: A Survey, *arXiv.* https://arxiv.org/pdf/2210.11831

[3] Aakash Sharma et al., (2024). GPU Cluster Scheduling for Network-Sensitive Deep Learning, *arXiv*. https://arxiv.org/pdf/2401.16492

[4] Jaewon Son et al., (2021). A GPU Scheduling Framework to Accelerate Hyper-Parameter Optimization in Deep Learning Clusters, *MDPI.* https://www.mdpi.com/2079-9292/10/3/350

[5] WEI GAO et al., (2022). Deep Learning Workload Scheduling in GPU Datacenters: Taxonomy, Challenges and Vision, *arXiv*. https://arxiv.org/pdf/2205.11913

[6] Aaron Harlap et al., (2018). PipeDream: Fast and Efficient Pipeline Parallel DNN Training, *arXiv*. https://arxiv.org/pdf/1806.03377

[7] Emre Karabulut et al., (2025). THEMIS: Time, Heterogeneity, and Energy Minded Scheduling for Fair Multi-Tenant Use in FPGAs, *arXiv*. https://arxiv.org/pdf/2404.00507

[8] Emna Baccour et al., (2022). Pervasive AI for IoT applications: A Survey on Resource-efficient Distributed Artificial Intelligence, *arXiv*. https://arxiv.org/pdf/2105.01798

[9] Alexander Ulanov et al., (2017). Modeling Scalability of Distributed Machine Learning, *arXiv*. https://arxiv.org/pdf/1610.06276

[10] Zakariya Ba Alawi, (2025). A Comparative Survey of PyTorch vs TensorFlow for Deep Learning: Usability, Performance, and Deployment Trade-offs, *arXiv*. https://arxiv.org/pdf/2508.04035