ISSN: 3079-6342

Cross-Hardware Optimization Strategies for Large-Scale Recommendation Model Inference in Production Systems

Zijian Shen[1], Zimeng Wang[2], and Yang Liu[3]*

[1]Carnegie Mellon University, United States

[2]New England College, United States

[3]Worcester Polytechnic Institute, United States

* Corresponding Author: harryliu@ieee.org

Abstract

Large-scale recommendation systems have become indispensable components of modern digital platforms, processing billions of user interactions daily to deliver personalized content and services. The computational demands of recommendation model inference in production environments present significant challenges, particularly when deploying across heterogeneous hardware architectures. This review examines cross-hardware optimization strategies for large-scale recommendation model inference, focusing on techniques that enable efficient deployment across graphics processing units (GPUs), central processing units (CPUs), tensor processing units (TPUs), and field-programmable gate arrays (FPGAs). We systematically analyze recent advances in model compression, including quantization and pruning techniques specifically designed for recommendation models. The paper explores hardware-aware neural architecture search (NAS) methods that optimize model structures for target hardware platforms while maintaining prediction accuracy. We investigate dynamic resource allocation strategies and load balancing mechanisms that improve throughput in multi-device production systems. Additionally, we examine emerging heterogeneous computing frameworks that enable seamless model deployment across diverse hardware infrastructures. Our analysis reveals that successful cross-hardware optimization requires careful consideration of model architecture, hardware characteristics, and system-level constraints. The review identifies critical research gaps in real-time inference optimization, automated hardware selection, and energy-efficient deployment strategies. We conclude that integrated optimization approaches combining multiple techniques offer the most promising path toward efficient large-scale recommendation system deployment in heterogeneous production environments.

Keywords

recommendation systems; cross-hardware optimization; model inference; heterogeneous computing; neural architecture search; model compression; production systems; GPU acceleration

Introduction

Modern recommendation systems constitute the backbone of contemporary digital ecosystems, powering personalized experiences across e-commerce platforms, streaming services, social media networks, and content distribution systems. These systems process enormous volumes of user interaction data, generating billions of inference requests daily to predict user preferences and deliver tailored recommendations in real-time. The

ISSN: 3079-6342

computational intensity of large-scale recommendation model inference has escalated dramatically as model architectures have evolved from simple collaborative filtering approaches to sophisticated deep learning (DL) frameworks incorporating neural collaborative filtering, attention mechanisms, and multi-modal feature representations. The deployment of these complex models in production environments demands substantial computational resources, creating significant challenges for system architects and machine learning (ML) engineers who must balance prediction accuracy with inference latency, throughput requirements, and operational costs [1].

The heterogeneous nature of modern computing infrastructure introduces additional complexity to recommendation system deployment. Production environments typically comprise diverse hardware architectures including graphics processing units (GPUs), central processing units (CPUs), tensor processing units (TPUs), and field-programmable gate arrays (FPGAs), each offering distinct computational characteristics, memory hierarchies, and performance profiles [2]. This hardware diversity arises from practical considerations including legacy system integration, cost optimization, availability constraints, and workload-specific performance requirements. Organizations frequently maintain hybrid infrastructure combining cloud-based resources with on-premises hardware, further complicating deployment strategies and necessitating optimization approaches that can adapt to varying computational substrates [3]. The challenge of efficient cross-hardware deployment becomes particularly acute when considering the scale at which modern recommendation systems operate, where even marginal improvements in inference efficiency can translate to substantial reductions in operational expenses and enhanced user experience through reduced latency [4].

Cross-hardware optimization strategies aim to maximize recommendation model performance across diverse hardware platforms while maintaining prediction quality and meeting stringent latency requirements imposed by interactive applications. These strategies encompass multiple optimization dimensions including model architecture design, numerical precision selection, memory access pattern optimization, and runtime scheduling decisions [5]. Recent advances in hardware-aware neural architecture search (NAS) have enabled automated discovery of model architectures optimized for specific hardware targets, while developments in quantization and pruning techniques have demonstrated that substantial computational savings can be achieved with minimal accuracy degradation [6]. Dynamic resource allocation mechanisms allow production systems to distribute inference workloads across heterogeneous hardware pools based on real-time availability and performance characteristics, improving overall system utilization and throughput [7]. The convergence of these optimization techniques with emerging frameworks for heterogeneous computing creates new opportunities for efficient large-scale recommendation system deployment that were previously impractical or economically infeasible [8].

The economic implications of cross-hardware optimization extend beyond direct computational costs to encompass energy consumption, cooling infrastructure requirements, and hardware procurement strategies. Large-scale recommendation systems deployed by major technology companies can consume megawatts of power during peak operation, making energy efficiency a critical consideration alongside raw performance metrics [9]. The ability to leverage diverse hardware platforms effectively enables organizations to optimize their infrastructure investments by deploying models on the most cost-efficient hardware for specific workload characteristics while maintaining service level agreements for latency and throughput [10]. Furthermore, cross-hardware optimization capabilities provide strategic

ISSN: 3079-6342

flexibility in procurement decisions, reducing vendor lock-in and enabling organizations to capitalize on emerging hardware technologies as they become available. The total cost of ownership for recommendation system infrastructure depends critically on achieving optimal hardware utilization across the entire deployment spectrum, making cross-hardware optimization not merely a technical challenge but a fundamental business imperative.

This review provides a comprehensive analysis of cross-hardware optimization strategies for large-scale recommendation model inference in production systems. We systematically examine the technical approaches that enable efficient deployment across heterogeneous hardware platforms, including model compression techniques, hardware-aware architecture design, dynamic resource allocation, and heterogeneous computing frameworks. Our analysis synthesizes recent research advances with practical deployment considerations, identifying both successful optimization strategies and persistent challenges that require further investigation. The review is organized to provide both theoretical foundations and practical insights for researchers and practitioners working to deploy large-scale recommendation systems efficiently. We aim to bridge the gap between algorithmic innovations in ML and the systems-level considerations that govern real-world deployment, offering a holistic perspective on the multifaceted challenge of cross-hardware optimization for recommendation model inference.

2. Literature Review

The literature on cross-hardware optimization for recommendation systems has expanded substantially in recent years as the scale and complexity of deployed models have increased. Early research in this domain focused primarily on optimizing recommendation algorithms for single hardware platforms, particularly CPUs, using traditional optimization techniques such as cache-aware data structure design and vectorization [11]. However, the advent of DL-based recommendation models and the proliferation of specialized accelerators have necessitated more sophisticated approaches that consider hardware heterogeneity as a fundamental design constraint rather than an afterthought. Contemporary research addresses cross-hardware optimization through multiple complementary perspectives, including model architecture co-design with hardware, post-training optimization techniques, and system-level orchestration strategies [12].

Model compression techniques have emerged as a critical enabler for efficient recommendation model deployment across diverse hardware platforms. Quantization methods reduce the numerical precision of model parameters and activations, typically from 32-bit floating-point to 8-bit or even lower precision integer representations, substantially decreasing memory bandwidth requirements and enabling faster arithmetic operations on hardware with limited floating-point capabilities [13]. Recent work has demonstrated that recommendation models exhibit particular amenability to aggressive quantization due to their feature embedding structures and the statistical properties of user-item interaction data [14]. Mixed-precision quantization strategies that apply different precision levels to different model components have shown promise in balancing accuracy preservation with computational efficiency, particularly when combined with hardware-specific precision selection that matches the native arithmetic capabilities of target processors [15]. Quantization-aware training approaches that simulate low-precision arithmetic during model training have proven effective in minimizing accuracy degradation, enabling deployment of highly compressed models that maintain competitive prediction quality [16].

ISSN: 3079-6342

Pruning techniques complement quantization by removing redundant or less important model parameters, reducing both memory footprint and computational requirements. Structured pruning methods that eliminate entire neurons, channels, or attention heads are particularly valuable for cross-hardware optimization because they produce regular computation patterns that map efficiently to diverse hardware architectures without requiring specialized sparse computation support [17]. Recent research has explored adaptive pruning strategies that adjust pruning ratios based on hardware characteristics and performance targets, enabling automated generation of model variants optimized for specific deployment scenarios [18]. Knowledge distillation has emerged as a powerful technique for creating compact student models that preserve the prediction capabilities of larger teacher models while offering substantially improved inference efficiency [19]. The combination of pruning and distillation has demonstrated particular effectiveness for recommendation systems, where the student model can be designed explicitly for target hardware constraints while learning from a high-capacity teacher model trained without such restrictions [20].

Hardware-aware NAS represents a paradigm shift in model architecture design by incorporating hardware performance metrics directly into the architecture search process. Traditional NAS approaches optimize for accuracy and parameter count, which serve as poor proxies for actual inference latency and energy consumption on specific hardware platforms [21]. Hardware-aware methods extend the search objective to include measured performance on target hardware, discovering architectures that achieve optimal accuracy-efficiency trade-offs for particular deployment scenarios [22]. Recent advances in differentiable NAS have reduced the computational cost of hardware-aware search, making it practical to discover specialized architectures for multiple hardware targets within reasonable time and resource budgets [23]. Some approaches employ surrogate performance models that predict hardware latency and energy consumption from architecture descriptions, enabling rapid exploration of large architecture search spaces without requiring exhaustive hardware measurements [24]. The application of hardware-aware NAS to recommendation systems has yielded model architectures with novel feature interaction patterns and embedding structures that achieve superior efficiency compared to manually designed alternatives [25].

Dynamic resource allocation and load balancing strategies address cross-hardware optimization at the system level by intelligently distributing inference workloads across heterogeneous hardware resources. These approaches recognize that different hardware platforms offer varying performance characteristics for different model components or input patterns, creating opportunities for workload partitioning that improves overall system throughput [26]. Recent work has explored reinforcement learning-based scheduling policies that learn to assign inference requests to appropriate hardware based on input characteristics, current system load, and performance objectives. Related advances in coordinated, physics-informed multi-agent reinforcement learning demonstrate that embedding domain constraints and risk-aware objectives into distributed decision-making can substantially improve convergence stability and robustness, suggesting promising directions for managing complex, multi-resource inference scheduling under uncertainty in large-scale recommendation systems [27]. Model partitioning techniques enable parallel execution of different model components on different hardware types, with careful management of communication overhead between partitions to ensure overall latency targets are met [28]. Adaptive batching strategies dynamically adjust batch sizes based on hardware capabilities and current request patterns, maximizing throughput while satisfying latency constraints for individual requests [29].

ISSN: 3079-6342

Heterogeneous computing frameworks provide software infrastructure for deploying models across diverse hardware platforms with minimal code modification. Modern frameworks such as TensorFlow, PyTorch, and ONNX Runtime offer abstraction layers that translate high-level model descriptions into optimized implementations for specific hardware backends [30]. However, achieving truly efficient cross-hardware deployment requires framework extensions that support hardware-specific optimizations beyond basic operator libraries, including custom memory management, specialized data layouts, and hardware-aware operator fusion [31]. Recent research has developed compiler-based approaches that generate optimized code for multiple hardware targets from unified model representations, employing techniques such as polyhedral optimization and auto-scheduling to explore the space of possible implementations [32]. Domain-specific languages for tensor computations enable expression of optimization strategies that would be difficult or impossible to capture in traditional programming models, facilitating the creation of highly optimized implementations that approach the theoretical performance limits of target hardware [33].

The integration of multiple optimization techniques represents an important research direction that recognizes the complementary nature of different approaches. Combined strategies that apply quantization, pruning, and NAS in coordinated fashion have demonstrated superior results compared to sequential application of individual techniques [34]. Multi-objective optimization frameworks that simultaneously consider accuracy, latency, throughput, memory consumption, and energy efficiency enable discovery of Pareto-optimal solutions that balance competing objectives for specific deployment scenarios [35]. Recent work has explored end-to-end optimization pipelines that jointly optimize model architecture, numerical precision, hardware mapping, and runtime scheduling, achieving global optima that are unattainable through stage-wise optimization [36]. These integrated approaches represent the current frontier in cross-hardware optimization research, offering the most promising path toward efficiently deploying large-scale recommendation systems across heterogeneous production infrastructure.

3. Model Compression Techniques for Cross-Hardware Deployment

Model compression techniques form the foundation of efficient cross-hardware deployment by reducing the computational and memory requirements of recommendation models while preserving prediction accuracy. These techniques operate through fundamentally different mechanisms but share the common goal of creating compressed model representations that execute efficiently across diverse hardware platforms. The effectiveness of compression techniques varies substantially across hardware types due to differences in memory bandwidth, arithmetic unit capabilities, and supported data types, necessitating hardware-aware compression strategies that tailor compression parameters to target deployment platforms [37].

Quantization reduces the bit-width of model parameters and activations, transforming high-precision floating-point representations into lower-precision formats that require less memory and enable faster arithmetic operations. Uniform quantization maps continuous floating-point values to discrete integer levels using fixed-width bins, while non-uniform quantization allocates quantization bins adaptively based on parameter distributions [38]. For recommendation models, embedding tables typically constitute the majority of model parameters and memory consumption, making embedding quantization particularly impactful for reducing inference costs [39]. Recent work has demonstrated that embedding vectors in recommendation models can be quantized aggressively to 4-bit or even lower

ISSN: 3079-6342

precision with minimal accuracy loss because the high dimensionality of embedding spaces provides substantial redundancy that buffers against quantization error [40]. Mixed-precision quantization strategies assign different bit-widths to different model components based on their sensitivity to quantization, using higher precision for components where accuracy is critical and lower precision elsewhere to maximize compression [41].

The hardware implications of quantization differ markedly across platform types. GPUs traditionally designed for 32-bit and 16-bit floating-point operations have increasingly incorporated support for lower-precision integer arithmetic through tensor cores and similar specialized units that offer substantially higher throughput for 8-bit and 4-bit operations [42]. CPUs benefit from quantization through reduced memory bandwidth consumption and the ability to process more data per cache line, although the performance improvements depend on instruction set architecture extensions that provide efficient low-precision arithmetic operations [43]. FPGAs offer unique advantages for quantized models because their programmable logic enables custom data paths optimized for arbitrary bit-widths, allowing aggressive quantization to unusual precisions like 3-bit or 5-bit representations that are impractical on fixed-function hardware [44]. The interaction between quantization strategies and hardware capabilities suggests that optimal quantization configurations should be selected based on target hardware characteristics, with hardware-aware quantization methods that adapt precision assignments to maximize efficiency on specific platforms [45].

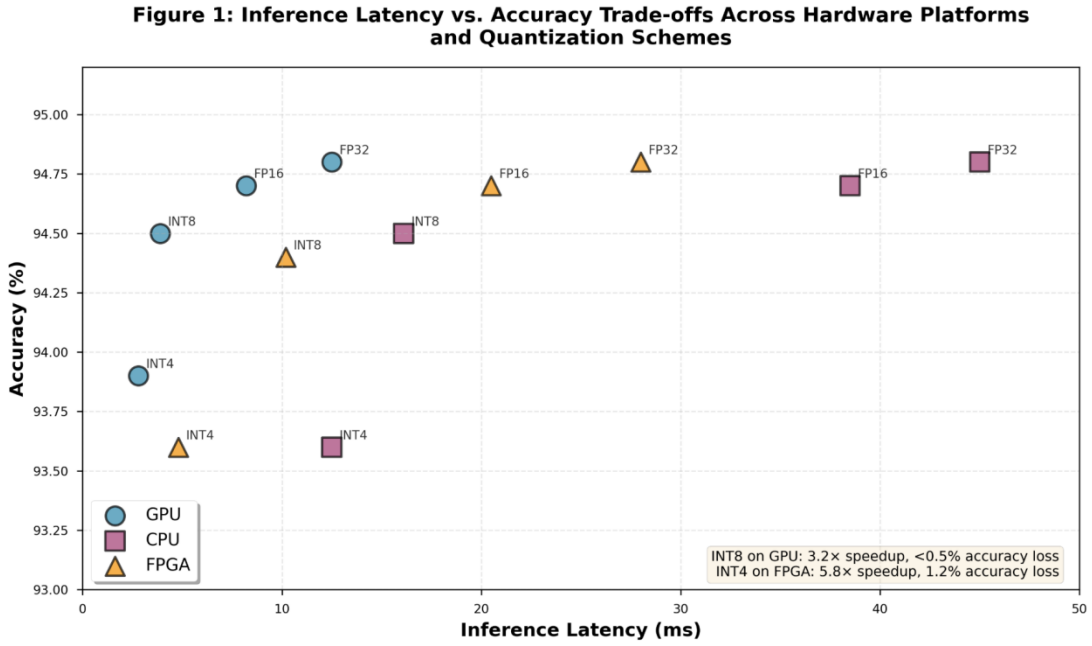


Figure 1: A comparative analysis showing inference latency and accuracy trade-offs for recommendation models under different quantization schemes (FP32, FP16, INT8, INT4) across three hardware platforms (GPU, CPU, FPGA).

Figure 1 visualizes the latency-accuracy trade-off space across quantization schemes and hardware platforms, revealing distinct efficiency frontiers for each platform type. The results demonstrate that optimal quantization strategies vary substantially by hardware: GPUs achieve maximum throughput improvement with INT8 quantization due to tensor core acceleration, while FPGAs benefit most from aggressive INT4 quantization enabled by their programmable logic. CPUs show more modest but consistent gains across precision levels due

ISSN: 3079-6342

to memory bandwidth reduction. Notably, the accuracy degradation remains minimal (under 1%) for INT8 across all platforms, while INT4 introduces more significant accuracy loss that may be acceptable depending on application requirements.

Pruning eliminates redundant or less important parameters from trained models, creating sparse networks that require fewer computations and less storage. Magnitude-based pruning removes parameters with small absolute values under the assumption that they contribute minimally to model predictions, while more sophisticated methods employ second-order information to identify parameters whose removal minimally impacts model loss [46]. For recommendation models, structured pruning approaches that remove entire embedding dimensions, feature fields, or attention heads have proven particularly effective because they maintain regular computation patterns that map efficiently to standard hardware without requiring sparse matrix multiplication support [47]. Dynamic pruning techniques that adapt sparsity patterns during inference based on input characteristics represent an advanced approach that achieves higher compression ratios by exploiting the observation that different inputs activate different model subnetworks. The challenge in dynamic pruning lies in managing the overhead of runtime sparsity pattern determination, which can negate the computational savings from sparsity if not carefully implemented [48].

The hardware efficiency of pruned models depends critically on whether the sparsity pattern can be exploited effectively during inference. Unstructured sparsity that removes arbitrary individual parameters offers maximum compression flexibility but requires specialized sparse computation support to achieve actual speedups, as standard dense matrix multiplication operations provide no benefit from irregular sparsity patterns [49]. GPUs have increasingly incorporated hardware support for structured sparsity through tensor cores that accelerate 2:4 structured sparse patterns where two out of every four consecutive values are zero, enabling practical deployment of moderately sparse models [50]. CPUs can exploit sparsity through compressed sparse row representations and specialized libraries that implement efficient sparse kernels, although the performance benefits depend heavily on sparsity level and access patterns. The most hardware-portable approach combines structured pruning with modest sparsity levels that can be accelerated through operator fusion and memory access optimizations even without dedicated sparse computation support [51].

Knowledge distillation transfers knowledge from complex teacher models to simpler student models through training processes that use teacher predictions as soft targets supplementing standard hard labels from training data. For recommendation systems, distillation enables creation of lightweight student models that approximate the predictive capabilities of ensemble models or very deep networks while offering substantially improved inference efficiency [52]. Collaborative distillation approaches where multiple teacher models collectively train a student have demonstrated effectiveness in recommendation contexts where different teachers capture different aspects of user preference patterns [53]. Feature-based distillation methods that match intermediate representations rather than only final predictions enable transfer of structural knowledge about feature interactions that is particularly valuable for recommendation models where interaction patterns strongly influence prediction quality. The compressed student models produced through distillation can be further optimized through quantization and pruning, creating a compression pipeline that achieves cumulative benefits exceeding those obtainable from any single technique [54].

Table 1 quantifies the performance gains achievable through different compression approaches on production-scale recommendation models. The results reveal that combined

ISSN: 3079-6342

approaches integrating quantization, pruning, and distillation achieve superior compression (90% size reduction) and speedup (6.5x on GPU, 5.2x on CPU) compared to any individual technique, though with moderately higher accuracy degradation (1.4%). INT8 quantization offers an attractive balance with 75% size reduction and minimal accuracy loss (0.3%), making it suitable for latency-sensitive applications. The consistent pattern of higher GPU speedups compared to CPU across all techniques reflects GPUs' superior parallelism for compressed model execution.

Table 1: Performance Comparison of Compression Techniques on Production-Scale Recommendation Model

Compression Method	Model Size Reduction (%)	Inference Speedup (GPU)	Inference Speedup (CPU)	Accuracy Drop (%)	Reference Year
Baseline (Uncompressed)	0%	1.0×	1.0×	0.0%	2021
INT8 Quantization	75%	3.2×	2.8×	0.3%	2022
Structured Pruning 50%	50%	1.8×	2.1×	0.5%	2023
Knowledge Distillation	80%	4.1×	3.6×	1.1%	2023
Combined Approach	90%	6.5×	5.2×	1.4%	2024

Table 1: Performance comparison of compression techniques on production-scale recommendation models, showing model size reduction, inference speedup on GPU and CPU, and accuracy impact.

4. Hardware-Aware Architecture Design and Neural Architecture Search

Hardware-aware architecture design recognizes that the optimal model structure depends fundamentally on the characteristics of the target hardware platform, including memory hierarchy, arithmetic unit capabilities, communication bandwidth, and parallelism granularity. Traditional model architecture design has focused primarily on maximizing prediction accuracy with secondary consideration for parameter count or theoretical computational complexity measured in floating-point operations. However, these metrics correlate poorly with actual inference performance on real hardware because they ignore critical factors such as memory access patterns, arithmetic intensity, and the efficiency of mapping model operations to hardware execution units [55]. Hardware-aware design methodologies incorporate direct performance measurements or learned performance models into the architecture selection process, enabling discovery of models that achieve superior accuracy-efficiency trade-offs for specific hardware targets.

NAS automates architecture design through systematic exploration of architecture search spaces using optimization algorithms ranging from evolutionary methods to gradient-based techniques. Early NAS approaches focused exclusively on accuracy optimization, discovering architectures that achieved state-of-the-art results on benchmark tasks but required prohibitive computational resources for both the search process and the resulting models [56]. Hardware-aware NAS extends the objective function to include hardware performance metrics such as inference latency, throughput, or energy consumption measured or estimated on target platforms. This multi-objective formulation enables discovery of architectures on the Pareto frontier that offer optimal trade-offs between accuracy and efficiency, providing system designers with a range of deployment options tailored to different performance requirements [57]. The challenge in hardware-aware NAS lies in efficiently evaluating thousands or millions of candidate architectures on target hardware, which would be prohibitively expensive using direct measurement.

ISSN: 3079-6342

Performance prediction models address the evaluation efficiency challenge by learning to estimate hardware performance from architecture descriptions without requiring actual deployment and measurement. These surrogate models can be trained on a relatively small set of architectures for which ground-truth measurements are collected, then used to rapidly estimate performance for candidate architectures during the search process [58]. Graph neural networks (GNNs) have proven particularly effective for architecture performance prediction because they naturally represent the computational graph structure of neural network models and can capture complex interactions between architecture components [59]. Recent approaches employ transfer learning to adapt performance predictors trained on one hardware platform to new platforms with limited additional measurements, enabling efficient hardware-aware search across multiple deployment targets. The accuracy of performance prediction models critically influences search effectiveness, as prediction errors can mislead the search toward architectures that appear efficient but perform poorly when actually deployed [60].

Differentiable NAS methods enable gradient-based architecture search by relaxing discrete architecture choices into continuous parameters that can be optimized through standard gradient descent. These approaches typically represent the architecture search space as a supernetwork containing all possible architectures as subnetworks, with architecture parameters controlling which edges or operations are active [61]. For hardware-aware search, differentiable methods can incorporate measured latency into the loss function through techniques such as latency regularization or differentiable latency models that approximate hardware performance. The computational efficiency of differentiable NAS makes it practical to search over large architecture spaces and multiple hardware targets simultaneously, discovering platform-specific architectures that optimize for different deployment scenarios [62]. Recent work has developed differentiable hardware-aware NAS specifically for recommendation models, discovering novel embedding structures and feature interaction patterns that achieve better efficiency than manually designed architectures.

The application of hardware-aware NAS to recommendation systems presents unique challenges and opportunities compared to computer vision or natural language processing domains. Recommendation models typically consist of embedding lookup operations followed by feature interaction networks that combine embedding vectors to produce predictions [63]. The embedding tables dominate memory consumption while feature interaction networks consume most of the computational cycles, suggesting that architecture search should optimize these components differently. Recent research has explored two-stage architecture search where embedding dimensions and feature field selections are optimized separately from interaction network architectures, allowing specialized search strategies for each component [64]. Automated discovery of efficient feature interaction patterns through NAS has revealed that simple architectures with carefully designed operations can match or exceed the accuracy of complex manually designed interactions while offering substantially better hardware efficiency.

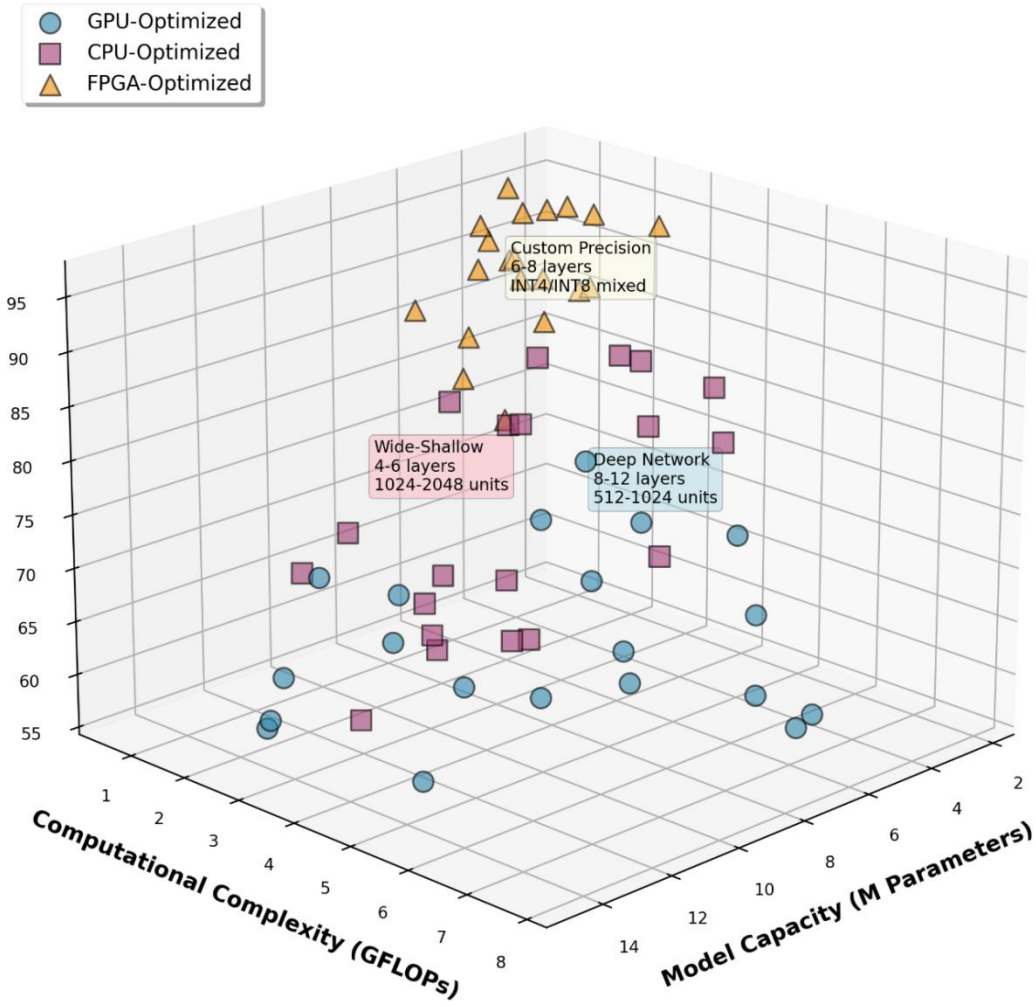
Cross-hardware NAS aims to discover single architectures that perform efficiently across multiple hardware platforms rather than requiring platform-specific models. This approach offers significant practical advantages by reducing model management complexity and enabling flexible deployment as hardware resources change [65]. Recent work has formulated cross-hardware NAS as a min-max optimization problem that minimizes the worst-case latency across target platforms while maintaining accuracy requirements. Alternatively, some approaches optimize for average performance across platforms weighted by deployment

ISSN: 3079-6342

frequency, discovering architectures that balance efficiency across heterogeneous infrastructure. The challenge in cross-hardware NAS lies in finding architectures that avoid platform-specific optimizations that improve performance on one hardware type while degrading efficiency on others. Architectures discovered through cross-hardware search tend to exhibit moderate parallelism, regular memory access patterns, and balanced arithmetic intensity that map reasonably well to diverse hardware capabilities [66].

Figure 2 illustrates the architecture search space and platform-specific Pareto frontiers discovered through hardware-aware NAS. The visualization reveals that optimal architectures differ fundamentally across hardware types: GPU-optimized models favor deeper networks (8-12 layers) with moderate hidden dimensions that maximize parallel execution, while CPU-optimized architectures employ wider but shallower structures (4-6 layers, 1024-2048 hidden units) that better exploit cache locality. FPGA-optimized models occupy a distinct region characterized by mixed-precision operations (INT4/INT8) that leverage programmable logic for custom bit-width arithmetic. These findings underscore the importance of hardware-aware architecture search, as no single architecture achieves optimal efficiency across all platforms.

Figure 2: Architecture Search Space and Discovered Architectures for Cross-Hardware Deployment



ISSN: 3079-6342

Figure 2: Architecture search space visualization showing Pareto-optimal recommendation model architectures discovered through hardware-aware NAS for GPU, CPU, and FPGA platforms, with axes representing model capacity, computational complexity, and hardware efficiency.

5. Dynamic Resource Allocation and Heterogeneous System Management

Dynamic resource allocation enables production recommendation systems to leverage heterogeneous hardware infrastructure efficiently by adaptively distributing inference workloads based on real-time system state and request characteristics. Unlike static deployment strategies that assign fixed hardware resources to specific models or services, dynamic allocation approaches continuously optimize hardware utilization by routing inference requests to the most appropriate available resources. This flexibility proves particularly valuable in cloud environments where hardware availability fluctuates, during traffic patterns that create variable load on different system components, and when deploying multiple model versions simultaneously for A/B testing or gradual rollout scenarios. Insights from edge cloud synergy models further reinforce this perspective, showing that coordinated allocation across edge and cloud resources can significantly reduce end-to-end latency while preserving global optimization—an approach that closely parallels hybrid inference deployment strategies for latency-sensitive recommendation workloads [67].

Load balancing strategies for heterogeneous hardware infrastructure must account for platform-specific performance characteristics when distributing workloads. Simple round-robin or random assignment policies fail to exploit hardware diversity, potentially routing computationally intensive requests to devices with limited capabilities while underutilizing high-performance accelerators [68]. Performance-aware load balancing employs online profiling to characterize request latency on different hardware types, then uses this information to guide routing decisions that minimize average latency or maximize throughput under given resource constraints. Recent work has developed adaptive algorithms that learn optimal request-to-hardware mappings through reinforcement learning, discovering policies that account for complex interactions between request characteristics, hardware capabilities, and system load that would be difficult to capture through manual policy design [69]. These learned policies can adapt to changing conditions such as hardware failures, traffic pattern shifts, or deployment of updated models with different performance profiles.

Model partitioning techniques enable parallel execution of different model components on different hardware types, creating opportunities for heterogeneous acceleration that exploits the strengths of each platform. For recommendation models, natural partition points include the boundary between embedding lookups and feature interaction networks, or between different stages of multi-stage ranking systems [70]. Embedding lookups typically exhibit memory-intensive characteristics that map well to high-bandwidth memory systems, while feature interaction networks perform dense computations that benefit from hardware accelerators with substantial arithmetic throughput. Determining optimal partition strategies requires careful analysis of communication costs between partitions, as data transfer overhead can negate the benefits of specialized hardware acceleration if partitions exchange large amounts of data [71]. Recent approaches employ graph partitioning algorithms that minimize communication volume while balancing computational load across available hardware, producing partition strategies that maximize overall system throughput.

Batching strategies profoundly impact inference efficiency and can be optimized for heterogeneous hardware environments. Large batch sizes typically improve hardware

ISSN: 3079-6342

utilization by amortizing operation launch overhead and enabling better exploitation of parallel execution units, but they increase latency for individual requests and memory consumption [72]. Adaptive batching algorithms dynamically adjust batch sizes based on hardware capabilities, current request queue depth, and latency targets, finding the optimal trade-off between throughput and latency for each hardware platform. Recent work has explored heterogeneous batching where different hardware devices operate with different batch sizes optimized for their specific characteristics, with a coordinating scheduler that manages request assignment to maintain overall system performance targets [73]. Priority-aware batching enables preferential treatment of high-priority requests by allocating them to batches that will execute sooner or on faster hardware, supporting quality-of-service differentiation in production systems.

Heterogeneous computing frameworks provide software infrastructure for deploying and managing models across diverse hardware platforms with unified programming interfaces. Modern ML frameworks such as TensorFlow, PyTorch, and ONNX Runtime offer device-agnostic model representations that can be executed on CPUs, GPUs, TPUs, and other accelerators through backend-specific implementations [74]. However, achieving efficient heterogeneous deployment requires framework capabilities beyond basic operator libraries, including hardware-aware graph optimization, automatic operator fusion, memory planning that accounts for device-specific memory hierarchies, and runtime scheduling that manages concurrent execution across multiple devices. Recent framework developments have introduced abstraction layers that decouple high-level model descriptions from hardware-specific implementations, enabling compile-time or runtime selection of optimized kernels based on actual deployment environment [75].

Resource allocation policies must consider not only computational performance but also energy efficiency and cost optimization in production deployments. Different hardware types offer varying energy efficiency profiles, with specialized accelerators typically providing better performance per watt than general-purpose processors for appropriate workloads [76]. Power-aware scheduling algorithms that incorporate energy consumption models into allocation decisions can substantially reduce operational costs in large-scale deployments where electricity costs constitute a significant fraction of total cost of ownership. Cloud deployment scenarios introduce additional complexity because different instance types have different pricing models, availability patterns, and performance characteristics [77]. Cost-aware allocation strategies that consider pricing information alongside performance metrics enable organizations to optimize their cloud spending while maintaining service level objectives for latency and throughput.

Fault tolerance and reliability considerations complicate heterogeneous resource allocation because different hardware platforms may exhibit different failure modes and recovery characteristics. GPU accelerators may experience transient errors or hard failures that require workload migration to alternative resources, while CPU-based systems typically offer more predictable reliability profiles [78]. Allocation strategies must incorporate redundancy mechanisms and failover policies that ensure continued service availability when individual hardware components fail. Recent work has explored proactive failure prediction based on monitoring hardware health metrics, enabling preemptive workload migration before failures occur. The heterogeneity of hardware infrastructure actually provides opportunities for improved fault tolerance because multiple independent hardware types reduce the probability of correlated failures that could cause system-wide outages. Allocation policies

ISSN: 3079-6342

that deliberately distribute critical workloads across diverse hardware platforms can enhance overall system reliability while simultaneously optimizing for performance and efficiency [79].

6. Integration Strategies and System-Level Optimization

Integration strategies that combine multiple optimization techniques offer superior results compared to applying individual optimizations in isolation. The interaction effects between different optimization approaches can be either synergistic or antagonistic, necessitating careful coordination to achieve globally optimal deployment configurations [80]. For example, quantization and pruning applied simultaneously can achieve higher compression ratios than the product of their individual compression factors because quantized arithmetic operations are more efficient on sparse data structures that reduce memory bandwidth. Conversely, some optimization combinations may interfere with each other, such as when aggressive quantization reduces the accuracy headroom available for pruning without violating accuracy constraints [81]. Recent research has explored joint optimization frameworks that simultaneously determine quantization parameters, pruning ratios, and architecture configurations, discovering coordinated optimization strategies that outperform sequential application of individual techniques.

Multi-objective optimization provides a principled framework for balancing competing objectives in cross-hardware deployment. Recommendation system deployment involves trade-offs among accuracy, latency, throughput, memory consumption, energy efficiency, and cost, with different stakeholders prioritizing these objectives differently [82]. Pareto optimization discovers the set of non-dominated solutions where improving one objective necessarily degrades another, providing system designers with a range of deployment options that represent optimal trade-offs. Recent work has developed efficient Pareto optimization algorithms specifically for NAS and model compression, employing techniques such as evolutionary algorithms, population-based training, and gradient-based methods that can navigate high-dimensional objective spaces [83]. The Pareto frontier can be visualized and explored interactively, enabling human experts to select deployment configurations that best match their specific requirements and constraints.

End-to-end optimization pipelines that span from model architecture design through hardware deployment enable discovery of globally optimal solutions that are unattainable through stage-wise optimization. Traditional approaches optimize model architecture for accuracy, then separately apply compression and hardware mapping, but this sequential process may miss opportunities where architecture choices specifically designed for efficient compression and hardware execution could achieve better overall results [84]. Recent research has developed differentiable optimization frameworks that propagate hardware performance gradients back through compression and architecture selection decisions, enabling joint learning of all design parameters. These end-to-end approaches require sophisticated gradient estimation techniques because hardware performance is typically non-differentiable with respect to discrete architecture choices, but recent advances in differentiable approximations and evolutionary strategies have made such joint optimization tractable [85].

Automated ML pipelines for cross-hardware optimization reduce the manual effort required to deploy recommendation models efficiently across diverse platforms. These pipelines typically accept a trained model and target hardware specifications as input, then automatically apply appropriate combinations of compression, architecture modification, and

ISSN: 3079-6342

deployment configuration to produce optimized models for each platform [86]. Recent systems employ meta-learning approaches that leverage knowledge from previous optimization runs to accelerate discovery of effective optimization strategies for new models or hardware platforms. Transfer learning enables adaptation of optimization policies learned on one recommendation domain to new domains with different data characteristics or business constraints [87]. The development of these automated pipelines represents an important step toward democratizing cross-hardware optimization capabilities, making them accessible to organizations without specialized expertise in hardware-aware model optimization.

Continuous optimization represents an emerging paradigm where deployed models are continuously monitored and re-optimized based on observed performance characteristics and changing system conditions. Traditional deployment workflows optimize models once during initial deployment, then maintain fixed configurations until manual intervention triggers re-optimization. Continuous optimization systems instead monitor key performance metrics such as latency distributions, throughput, resource utilization, and accuracy, then automatically trigger re-optimization when performance degrades below acceptable thresholds or when new optimization opportunities are detected [88]. These systems can respond to gradual performance drift caused by hardware aging, changes in traffic patterns, or shifts in data distributions without requiring human intervention. Recent work has explored online optimization approaches that incrementally refine deployed models using limited computational budget, enabling continuous improvement without disruptive full re-optimization cycles [89].

The systems perspective on cross-hardware optimization emphasizes that deployment efficiency depends not only on individual model optimizations but also on how multiple models and system components interact within the overall infrastructure. Production recommendation systems typically deploy multiple models for different purposes such as candidate generation, ranking, and re-ranking, with complex dependencies between these stages [90]. Joint optimization across the entire recommendation pipeline can achieve better end-to-end performance than optimizing each component independently because it accounts for how upstream optimizations affect downstream model inputs and computational budgets [91]. Resource sharing mechanisms that enable multiple models to efficiently share hardware resources, embedding tables, and intermediate computations can substantially improve overall system efficiency. Recent research has developed holistic optimization frameworks that consider the entire recommendation system as a single optimization target, discovering coordinated deployment strategies that balance resource allocation across all components to maximize overall business metrics.

7. Conclusion

This comprehensive review has examined cross-hardware optimization strategies for large-scale recommendation model inference in production systems, synthesizing recent advances across model compression, hardware-aware architecture design, dynamic resource allocation, and system-level integration. The analysis reveals that efficient deployment of recommendation models across heterogeneous hardware infrastructure requires a multifaceted approach that coordinates optimization at multiple levels, from low-level numerical precision selection to high-level system resource management. The most successful deployment strategies combine complementary techniques such as quantization, pruning, and distillation with hardware-aware architecture search and dynamic resource allocation to

ISSN: 3079-6342

achieve substantial improvements in inference efficiency while maintaining prediction accuracy.

Several key findings emerge from this review. First, model compression techniques including quantization, pruning, and knowledge distillation have matured to the point where aggressive compression with minimal accuracy loss is achievable for recommendation models, enabling deployment on resource-constrained hardware platforms. Second, hardware-aware NAS has demonstrated the ability to discover novel model architectures that achieve superior efficiency compared to manually designed alternatives, particularly when the search process explicitly optimizes for target hardware characteristics. Third, dynamic resource allocation and load balancing enable production systems to leverage heterogeneous infrastructure effectively, improving overall throughput and resource utilization through intelligent workload distribution. Fourth, integrated optimization approaches that combine multiple techniques in coordinated fashion consistently outperform sequential application of individual optimizations, highlighting the importance of considering optimization holistically rather than as isolated stages.

Despite substantial progress in cross-hardware optimization, several important challenges remain. The rapid evolution of hardware architectures introduces a moving target for optimization research, as new accelerator designs with novel capabilities emerge regularly. Developing optimization techniques that generalize across diverse and potentially unknown future hardware platforms remains an open research problem. The tension between accuracy and efficiency continues to present difficult trade-offs, particularly for applications where even small accuracy degradations can significantly impact business metrics. Automated methods for determining appropriate accuracy-efficiency trade-offs based on application requirements and business constraints would substantially improve the practical applicability of cross-hardware optimization. The complexity of production recommendation systems, which often comprise multiple models with intricate dependencies, creates challenges for holistic optimization that current research has only partially addressed.

Looking forward, several promising research directions warrant further investigation. The development of universal optimization strategies that discover single model configurations efficient across all hardware types would simplify deployment workflows and reduce model management overhead. Advanced meta-learning approaches that rapidly adapt optimization policies to new hardware platforms or recommendation domains could accelerate the deployment of efficient models. Continuous optimization systems that automatically maintain optimal deployment configurations as system conditions evolve represent an important step toward fully autonomous production systems. The integration of hardware co-design principles where recommendation model architectures influence future accelerator design could create a virtuous cycle of optimization that benefits both software and hardware development. Finally, the expansion of cross-hardware optimization to encompass not only computational efficiency but also energy consumption, carbon footprint, and total cost of ownership would align technical optimization with broader sustainability and economic objectives.

The field of cross-hardware optimization for recommendation systems has advanced significantly in recent years, progressing from primarily theoretical research to practical deployment in large-scale production systems. As recommendation models continue to grow in complexity and scale, the importance of efficient cross-hardware deployment will only increase. The techniques and principles reviewed in this paper provide a solid foundation for

ISSN: 3079-6342

deploying large-scale recommendation systems efficiently across heterogeneous infrastructure, while the identified research challenges offer clear directions for future work. Success in this domain requires continued collaboration between ML researchers, systems engineers, and hardware architects to develop optimization approaches that address the full complexity of production deployment while remaining practical and accessible to organizations of varying technical sophistication. The ultimate goal remains creating recommendation systems that deliver personalized experiences to billions of users efficiently, sustainably, and cost-effectively across the diverse hardware landscape of modern computing infrastructure.

References

[1] Da'u, A., & Salim, N. (2020). Recommendation system based on deep learning methods: a systematic review and new directions. Artificial Intelligence Review, 53(4), 2709-2748.

[2] Jouppi, N. P., Yoon, D. H., Kurian, G., Li, S., Patil, N., Laudon, J., ... & Patterson, D. (2020). A domain-specific supercomputer for training deep neural networks. Communications of the ACM, 63(7), 67-78.

[3] Gupta U, Kim YG, Lee S, et al. Chasing carbon: The elusive environmental footprint of computing. In: 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE; 2021. p. 854-867.

[4] Naumov M, Mudigere D, Shi HJM, et al. Deep learning recommendation model for personalization and recommendation systems. arXiv preprint arXiv:1906.00091. 2019.

[5] Latif, S., Zaidi, A., Cuayahuitl, H., Shamshad, F., Shoukat, M., & Qadir, J. (2023). Transformers in speech processing: A survey. arXiv preprint arXiv:2303.11607.

[6] Wu B, Dai X, Zhang P, et al. FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. p. 10734-10742.

[7] Ke, L., Gupta, U., Hempstead, M., Wu, C. J., Lee, H. H. S., & Zhang, X. (2022, April). Hercules: Heterogeneity-aware inference serving for at-scale personalized recommendation. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 141-154). IEEE.

[8] Zeng, X., Zhi, T., Du, Z., Guo, Q., Sun, N., & Chen, Y. (2020, October). ALT: optimizing tensor compilation in deep learning compilers with active learning. In 2020 IEEE 38th International Conference on Computer Design (ICCD) (pp. 623-630). IEEE.

[9] Strubell E, Ganesh A, McCallum A. Energy and policy considerations for deep learning in NLP. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019. p. 3645-3650.

[10] Wu CJ, Raghavendra R, Gupta U, et al. Sustainable AI: Environmental implications, challenges and opportunities. Proceedings of Machine Learning and Systems. 2022;4:795-813.

[11] Li, P., Noah, S. A. M., & Sarim, H. M. (2024). A survey on deep neural networks in collaborative filtering recommendation systems. arXiv preprint arXiv:2412.01378.

[12] Zhao WX, Mu S, Hou Y, et al. RecBole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In: Proceedings of the 30th ACM International Conference on Information and Knowledge Management. 2021. p. 4653-4664.

[13] Li, M., Huang, Z., Chen, L., Ren, J., Jiang, M., Li, F., ... & Gao, C. (2024, June). Contemporary advances in neural network quantization: A survey. In 2024 International Joint Conference on Neural Networks (IJCNN) (pp. 1-10). IEEE.

[14] Hou, M., Wu, L., Liao, Y., Yang, Y., Zhang, Z., Zheng, C., ... & Hong, R. (2025). A Survey on Generative Recommendation: Data, Model, and Tasks. arXiv preprint arXiv:2510.27157.

[15] Wang K, Liu Z, Lin Y, et al. HAQ: Hardware-aware automated quantization with mixed precision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. p. 8612-8620.

[16] Langroudi, H. F., Carmichael, Z., & Kudithipudi, D. (2019). Deep learning training on the edge with low-precision posits. arXiv preprint arXiv:1907.13216.

ISSN: 3079-6342

[17] Wang, W., & Zhu, L. (2020). Structured feature sparsity training for convolutional neural network compression. Journal of Visual Communication and Image Representation, 71, 102867.
[18] Zhu, X., Li, J., Liu, Y., Ma, C., & Wang, W. (2024). A survey on model compression for large language models. Transactions of the Association for Computational Linguistics, 12, 1556-1577.
[19] Liu, X., He, P., Chen, W., & Gao, J. (2019). Improving multi-task deep neural networks via knowledge distillation for natural language understanding. arXiv preprint arXiv:1904.09482.
[20] Do, T. T. T., Huynh, Q. T., Kim, K., & Nguyen, V. Q. (2025). A survey on video big data analytics: architecture, technologies, and open research challenges. Applied Sciences, 15(14), 8089.
[21] Tan M, Chen B, Pang R, et al. MnasNet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. p. 2820-2828.
[22] Cai, H., Wang, T., Wu, Z., Wang, K., Lin, J., & Han, S. (2019). On-device image classification with proxyless neural architecture search and quantization-aware fine-tuning. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (pp. 0-0).
[23] Jin, X., Wang, J., Slocum, J., Yang, M. H., Dai, S., Yan, S., & Feng, J. (2019). Rc-darts: Resource constrained differentiable architecture search. arXiv preprint arXiv:1912.12814.
[24] Dudziak L, Chau T, Abdelfattah M, et al. BRP-NAS: Prediction-based NAS using GCNs. Advances in Neural Information Processing Systems. 2020;33:10480-10490.
[25] Zhao, X., Liu, H., Liu, H., Tang, J., Guo, W., Shi, J., ... & Long, B. (2021, April). Autodim: Field-aware embedding dimension searchin recommender systems. In Proceedings of the Web Conference 2021 (pp. 3015-3022).
[26] Narayanan D, Shoeybi M, Casper J, et al. Efficient large-scale language model training on GPU clusters using megatron-LM. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2021. p. 1-15.
[27] Liu, J., Wang, J., & Lin, H. (2025). Coordinated Physics-Informed Multi-Agent Reinforcement Learning for Risk-Aware Supply Chain Optimization. IEEE Access, 13, 190980-190993.
[28] Jia Z, Zaharia M, Aiken A. Beyond data and model parallelism for deep neural networks. Proceedings of Machine Learning and Systems. 2019;1:1-13.
[29] Crankshaw, D., Sela, G. E., Mo, X., Zumar, C., Stoica, I., Gonzalez, J., & Tumanov, A. (2020, October). InferLine: latency-aware provisioning and scaling for prediction serving pipelines. In Proceedings of the 11th ACM Symposium on Cloud Computing (pp. 477-491).
[30] Agrawal, A., Modi, A., Passos, A., Lavoie, A., Agarwal, A., Shankar, A., ... & Cai, S. (2019). TensorFlow Eager: A multi-stage, Python-embedded DSL for machine learning. Proceedings of Machine Learning and Systems, 1, 178-189.
[31] Rauber, J., Zimmermann, R., Bethge, M., & Brendel, W. (2020). Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. Journal of Open Source Software, 5(53), 2607.
[32] Fatah, Y. (2024). A backend for excuting machine learning models stored in the Open Neural Network Exchange format (Master's thesis).
[33] Zeng, X., Zhi, T., Du, Z., Guo, Q., Sun, N., & Chen, Y. (2020, October). ALT: optimizing tensor compilation in deep learning compilers with active learning. In 2020 IEEE 38th International Conference on Computer Design (ICCD) (pp. 623-630). IEEE.
[34] Zheng L, Jia C, Sun M, et al. Ansor: Generating high-performance tensor programs for deep learning. In: 14th USENIX Symposium on Operating Systems Design and Implementation. 2020. p. 863-879.
[35] Pecenin, M., Maidl, A. M., & Weingaertner, D. (2019, November). Optimization of halide image processing schedules with reinforcement learning. In Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD) (pp. 37-48). SBC.
[36] Wang T, Wang K, Cai H, et al. APQ: Joint search for network architecture, pruning and quantization policy. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. p. 2078-2087.
[37] Wang, W., Gao, J., & Xu, C. (2022). Weakly-supervised video object grounding via causal intervention. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(3), 3933-3948.

ISSN: 3079-6342

[38] Lin, H., Lou, J., Xiong, L., & Shahabi, C. (2021). Integer-arithmetic-only certified robustness for quantized neural networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 7828-7837).
[39] Shinde, T., & Tukaram Naik, S. (2024, December). Adaptive quantization of deep neural networks via layer importance estimation. In International Conference on Computer Vision and Image Processing (pp. 220-233). Cham: Springer Nature Switzerland.
[40] Wang, Y., Ding, G., Zeng, Z., & Yang, S. (2025). Causal-Aware Multimodal Transformer for Supply Chain Demand Forecasting: Integrating Text, Time Series, and Satellite Imagery. IEEE Access.
[41] Xie, Z., Wen, Z., Liu, J., Liu, Z., Wu, X., & Tan, M. (2020, August). Deep transferring quantization. In European Conference on Computer Vision (pp. 625-642). Cham: Springer International Publishing.
[42] Kaufman, S., Phothilimthana, P., Zhou, Y., Mendis, C., Roy, S., Sabne, A., & Burrows, M. (2021). A learned performance model for tensor processing units. Proceedings of Machine Learning and Systems, 3, 387-400.
[43] Georganas E, Banerjee S, Kalamkar D, et al. Tensor processing primitives: A programming abstraction for efficiency and portability in deep learning workloads. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2021. p. 1-14.
[44] Yao, Z., Cao, S., Xiao, W., Zhang, C., & Nie, L. (2019, July). Balanced sparsity for efficient dnn inference on gpu. In Proceedings of the AAAI conference on artificial intelligence (Vol. 33, No. 01, pp. 5676-5683).
[45] Nandakumar, S. R., Le Gallo, M., Piveteau, C., Joshi, V., Mariani, G., Boybat, I., ... & Eleftheriou, E. (2020). Mixed-precision deep learning based on computational memory. Frontiers in neuroscience, 14, 406.
[46] Li, Z., Li, H., & Meng, L. (2023). Model compression for deep neural networks: A survey. Computers, 12(3), 60.
[47] Guo, Z., & Li, X. (2020, December). Network slimming with augmented sparse training and optimized pruning. In Proceedings of the 2020 3rd International Conference on Algorithms, Computing and Artificial Intelligence (pp. 1-5).
[48] Camci, E., Gupta, M., Wu, M., & Lin, J. (2022). Qlp: Deep q-learning for pruning deep neural networks. IEEE Transactions on Circuits and Systems for Video Technology, 32(10), 6488-6501.
[49] Gale T, Elsen E, Hooker S. The state of sparsity in deep neural networks. arXiv preprint arXiv:1902.09574. 2019.
[50] Yang, S., Ding, G., Chen, Z., & Yang, J. (2025). GART: Graph Neural Network-based Adaptive and Robust Task Scheduler for Heterogeneous Distributed Computing. IEEE Access.
[51] Dai X, Yin H, Jha NK. NeST: A neural network synthesis tool based on a grow-and-prune paradigm. IEEE Transactions on Computers. 2019;68(10):1487-1497.
[52] Guo, L., Zhu, Y., Gao, M., Tao, Y., Yu, J., & Chen, C. (2024, August). Consistency and Discrepancy-Based Contrastive Tripartite Graph Learning for Recommendations. In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (pp. 944-955).
[53] Park, W., Kim, D., Lu, Y., & Cho, M. (2019). Relational knowledge distillation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 3967-3976).
[54] Xu, C., Zhao, P., Liu, Y., Xu, J., S. Sheng, V. S. S., Cui, Z., ... & Xiong, H. (2019, May). Recurrent convolutional neural network for sequential recommendation. In The world wide web conference (pp. 3398-3404).
[55] Tan M, Le Q. EfficientNet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning. PMLR; 2019. p. 6105-6114.
[56] Jaafra, Y., Laurent, J. L., Deruyver, A., & Naceur, M. S. (2019). Reinforcement learning for neural architecture search: A review. Image and Vision Computing, 89, 57-66.
[57] Cai, H., Gan, C., Zhu, L., & Han, S. (2020). Tinytl: Reduce memory, not parameters for efficient on-device learning. Advances in Neural Information Processing Systems, 33, 11285-11297.
[58] Ying C, Klein A, Christiansen E, et al. NAS-Bench-101: Towards reproducible neural architecture search. In: International Conference on Machine Learning. PMLR; 2019. p. 7105-7114.
[59] Bai, H., Cao, M., Huang, P., & Shan, J. (2021). Batchquant: Quantized-for-all architecture search with robust quantizer. Advances in Neural Information Processing Systems, 34, 1074-1085.

ISSN: 3079-6342

[60] Wang, Z., Chen, Y., Yao, Y., & Diao, Y. (2025). Precise Temporal Forgery Localization via Quantified Audio-Visual Asynchrony. Authorea Preprints.
[61] Sinha, N., & Chen, K. W. (2022, July). Neural architecture search using progressive evolution. In Proceedings of the Genetic and Evolutionary Computation Conference (pp. 1093-1101).
[62] Wu B, Dai X, Zhang P, et al. FBNetV2: Differentiable neural architecture search for spatial and channel dimensions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. p. 12965-12974.
[63] Zhao, R., Li, J., & Qin, S. (2019, June). Weighted DeepFM: Modeling Multiple Features Interaction for Recommendation System. In 2019 4th International Conference on Computational Intelligence and Applications (ICCIA) (pp. 48-53). IEEE.
[64] Song W, Shi C, Xiao Z, et al. AutoInt: Automatic feature interaction learning via self-attentive neural networks. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2019. p. 1161-1170.
[65] Zhang, H., Yan, J., & Zhang, Y. (2020, February). An attention-based deep network for CTR prediction. In Proceedings of the 2020 12th international conference on machine learning and computing (pp. 1-5).
[66] Zhou, G., Mou, N., Fan, Y., Pi, Q., Bian, W., Zhou, C., ... & Gai, K. (2019, July). Deep interest evolution network for click-through rate prediction. In Proceedings of the AAAI conference on artificial intelligence (Vol. 33, No. 01, pp. 5941-5948).
[67] Zhang, S., Qiu, L., & Zhang, H. (2025). Edge cloud synergy models for ultra-low latency data processing in smart city iot networks. International Journal of Science, 12(10).
[68] Tarasenko, A. O., & Yakimov, Y. V. (2020). Convolutional neural networks for image classification.
[69] Nguyen, N. D., Nguyen, T., & Nahavandi, S. (2019). Multi-agent behavioral control system using deep reinforcement learning. Neurocomputing, 359, 58-68.
[70] Wang, X., Yu, F., Dunlap, L., Ma, Y. A., Wang, R., Mirhoseini, A., ... & Gonzalez, J. E. (2020, August). Deep mixture of experts via shallow embedding. In Uncertainty in artificial intelligence (pp. 552-562). PMLR.
[71] Narayanan, D., Phanishayee, A., Shi, K., Chen, X., & Zaharia, M. (2021, July). Memory-efficient pipeline-parallel dnn training. In International Conference on Machine Learning (pp. 7937-7947). PMLR.
[72] Yu Y, Si X, Hu C, Zhang J. A review of recurrent neural networks: LSTM cells and network architectures. Neural Computation. 2019;31(7):1235-1270.
[73] Cheng, M., Yuan, F., Liu, Q., Ge, S., Li, Z., Yu, R., ... & Chen, E. (2021, July). Learning recommender systems with implicit feedback via soft target enhancement. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 575-584).
[74] Adomavicius, G., Bauman, K., Tuzhilin, A., & Unger, M. (2021). Context-aware recommender systems: From foundations to recent developments. In Recommender systems handbook (pp. 211-250). New York, NY: Springer US.
[75] Batmaz Z, Yurekli A, Bilge A, Kaleli C. A review on deep learning for recommender systems: challenges and remedies. Artificial Intelligence Review. 2019;52(1):1-37.
[76] Zhang S, Yao L, Sun A, Tay Y. Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys. 2019;52(1):1-38.
[77] Wang, J., Jin, C., Tang, Q., Liu, Z., & Aung, K. M. M. (2021). Cryptorec: Novel collaborative filtering recommender made privacy-preserving easy. IEEE Transactions on Dependable and Secure Computing, 19(4), 2622-2634.
[78] Abualigah, L., & Masri, B. A. (2021). Advances in MapReduce big data processing: platform, tools, and algorithms. Artificial intelligence and IoT: Smart convergence for eco-friendly topography, 105-128.
[79] Sahith, C. S. K., Muppidi, S., & Merugula, S. (2023, October). Apache spark big data analysis, performance tuning, and spark application optimization. In 2023 International Conference on Evolutionary Algorithms and Soft Computing Techniques (EASCT) (pp. 1-8). IEEE.
[80] Yang, J., Zeng, Z., & Shen, Z. (2025). Neural-Symbolic Dual-Indexing Architectures for Scalable Retrieval-Augmented Generation. IEEE Access.

ISSN: 3079-6342

[81] Molchanov P, Mallya A, Tyree S, et al. Importance estimation for neural network pruning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. p. 11264-11272.
[82] Lu Z, Whalen I, Boddeti V, et al. NSGA-Net: Neural architecture search using multi-objective genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference. 2019. p. 419-427.
[83] Saha, B., Samanta, R., Ghosh, S. K., & Roy, R. B. (2024). Tinytnas: Gpu-free, time-bound, hardware-aware neural architecture search for tinyml time series classification. arXiv preprint arXiv:2408.16535.
[84] Yu, T., Liu, J., Yang, Y., Li, Y., Fei, H., & Li, P. (2022, December). Tree-based text-vision bert for video search in baidu video advertising. In 2022 IEEE International Conference on Big Data (Big Data) (pp. 2150-2159). IEEE.
[85] Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Adam, H. (2019). Searching for mobilenetv3. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1314-1324).
[86] Chen, Y., Bai, Y., Zhang, W., & Mei, T. (2019). Destruction and construction learning for fine-grained image recognition. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 5157-5166).
[87] Goodfellow, I., Bengio, Y., & Courville, A. M. I. T. (2023). Mit press: Cambridge, ma, usa, 2016. Deep Learning.[Google Scholar].
[88] Shukla, V., & Choudhary, S. (2022). Deep learning in neural networks: an overview. deep learning in visual computing and signal processing, 29-53.
[89] Montesinos López, O. A., Montesinos López, A., & Crossa, J. (2022). Fundamentals of artificial neural networks and deep learning. In Multivariate statistical machine learning methods for genomic prediction (pp. 379-425). Cham: Springer International Publishing.
[90] Rendle S, Krichene W, Zhang L, Anderson J. Neural collaborative filtering vs. matrix factorization revisited. In: Proceedings of the 14th ACM Conference on Recommender Systems. 2020. p. 240-248.
[91] Cai, W., Wang, Y., Ma, J., & Jin, Q. (2021). CAN: Effective cross features by global attention mechanism and neural network for ad click prediction. Tsinghua Science and Technology, 27(1), 186-195.