10.48047/jocaaa.2025.34.12.30

Edge AI Inference Optimization: Quantization and Pruning on Resource-Constrained Platforms

Ishan Pardesi
Carnegie Mellon University, USA

Abstract

The deployment of sophisticated artificial intelligence models on resource-constrained embedded systems presents fundamental challenges in balancing computational efficiency with accuracy preservation. Contemporary edge devices, including ARM Cortex-A processors and automotive electronic control units, operate under severe limitations of memory capacity, computational throughput, and power budgets that preclude direct deployment of standard floating-point neural networks. Quantization techniques systematically reduce numerical precision from 32-bit floating-point to 8-bit or binary integer representations, achieving compression ratios exceeding 50× while maintaining accuracy within acceptable degradation thresholds. Post-training quantization enables direct conversion of pre-trained models without retraining requirements, while quantization-aware training adapts network representations to accommodate extreme precision reduction. Pruning methodologies exploit overparameterization in neural architectures through selective parameter elimination, with magnitude-based approaches achieving sparsity levels of 80-90% and structured pruning variants enabling hardware acceleration on conventional processors. Hardware-aware optimization strategies align sparsity patterns with SIMD execution units and memory access characteristics, maximizing inference throughput on embedded platforms. Empirical validation in ARM Cortex-A processors and Raspberry Pi systems demonstrates practical deployment of vision and language models within tight resource envelopes, achieving previously unattainable real-time inference performance on cost-effective embedded hardware. The convergence of efficient neural architecture, aggressive model compression, and platform-specific optimization enables the democratization of artificial intelligence capabilities in value-sensitive applications in the automotive, industrial, and consumer domains.

Keywords: Edge AI Optimization, Neural Network Quantization, Model Pruning, Embedded Deep Learning, Resource-Constrained Inference

1. Introduction

The rise of embedded systems applications of artificial intelligence has generated a never-before demand for effective model deployment techniques, combining complex computational demands with extreme hardware limitations. Contemporary edge devices—such as ARM Cortex-A processors, automotive electronic control units (ECUs), and dedicated artificial intelligence accelerators—have tight constraints on memory capacity (usually under Power budget, computational throughput, and 4 GB RAM. These limitations call for basic changes in how neural networks are shown and run beyond conventional optimization techniques created for datacenter-scale infrastructure.

The fundamental challenge lies in maintaining model accuracy while achieving the dramatic resource reductions required for practical embedded deployment. Standard neural networks trained in floating-point precision (FP32) consume memory and computational resources that exceed the capabilities of cost-effective edge hardware by orders of magnitude. Recent advances in network compression through dynamic surgical pruning have demonstrated remarkable potential for addressing these resource constraints. The Dynamic Network Surgery approach, which combines both pruning and splicing

10.48047/jocaaa.2025.34.12.30

operations during training, has achieved compression ratios of 17.7× on AlexNet while maintaining top-5 accuracy within 0.11% of the baseline model on ImageNet classification tasks [1]. This technique extends beyond traditional magnitude-based pruning by incorporating connection importance estimation and allowing previously pruned connections to be restored during training when beneficial. Using this in the VGG-16 network, 10.5x compression was reached with a loss in accuracy of 0.16%, lowering the model size by 138 million parameters to about 13 million parameters without affecting the architectural depth or representativeness [1]. This resource imbalance has traditionally constrained advanced AI applications and services to cloud computing infrastructures, restricting real-time applications to autonomous systems, industrial automation, or consumer Internet of Things devices where latency, privacy, or connectivity constraints do not allow remote computing.

The architecture of edge computing has its basic principles in facilitating the solution to these deployment issues through the inference computations that are performed on the resource-constrained devices instead of being transferred to a remote cloud infrastructure, where the raw sensor information is received. The DeepX framework, specifically designed for mobile deep learning acceleration, demonstrated that optimized model execution on smartphone-class processors can achieve 7-day continuous operation for audio-based activity recognition tasks while consuming only 0.43% of battery capacity per hour [2]. This software accelerator framework implements resource management strategies, including heterogeneous layer-wise distribution across CPU and GPU cores, achieving 90ms inference latency for GoogleNet on Samsung Galaxy S6 devices with only 0.8W power consumption [2]. The framework's runtime scheduling reduces memory footprint by 32% through intelligent layer decomposition and execution pipelining, enabling deployment of networks requiring 50MB of parameters on devices with limited RAM availability.

This article examines comprehensive methodologies for neural network compression through quantization and pruning techniques specifically engineered for resource-constrained platforms. Through empirical validation across representative hardware platforms, the efficacy of these techniques is established in achieving substantial resource reductions while preserving functional accuracy requirements for production applications in embedded artificial intelligence systems.

10.48047/jocaaa.2025.34.12.30

Optimization Framework	Core Technique	Target Platform	Key Innovation	Performance Characteristic
Dynamic Network Surgery	Pruning and Splicing	General Deep Networks	Connection Restoration	Maintains Architectural Depth
DeepX Accelerator	Heterogeneous Scheduling	Mobile Devices	Layer-wise Distribution	Extended Battery Operation
Surgical Compression	Importance Estimation	Embedded Systems	Adaptive Parameter Removal	Minimal Accuracy Loss
Mobile Framework	Resource Management	Smartphone Processors	CPU-GPU Coordination	Low Power Consumption

Table 1: Edge AI Optimization Frameworks and Deployment Strategies [1, 2]

2. Quantization Methodologies for Embedded Neural Network Deployment

Quantization represents a fundamental approach to model compression wherein the numerical precision of network parameters and activations is systematically reduced from floating-point to lower-bitwidth integer representations. This transformation yields proportional reductions in memory footprint and computational complexity while introducing controlled approximation error that must be carefully managed to preserve model accuracy.

Post-training quantization (PTQ) offers the most accessible pathway for model compression, converting pre-trained FP32 models to INT8 representation without requiring additional training procedures. Comprehensive analysis of quantization techniques has demonstrated that INT8 quantization applied to ResNet-50 achieves only 0.7% accuracy loss on ImageNet classification, declining from 76.1% top-1 accuracy in FP32 to 75.4% in INT8, while reducing model size from 102MB to 25.5MB, representing a 4× compression ratio [3]. This technique achieves a 75% reduction in memory requirements through direct numerical conversion, eliminating floating-point hardware dependencies that significantly increase silicon area and power consumption in embedded processors. The calibration process involves analyzing activation distributions across representative input datasets to determine optimal scaling factors for each quantized layer. Per-channel quantization schemes refine this approach by computing independent scaling parameters for individual convolutional channels, thereby minimizing quantization error across layers with heterogeneous activation ranges. Empirical validation on MobileNet-V1 demonstrates that per-channel quantization maintains 70.9% top-1 accuracy compared to 70.6% baseline FP32 performance, essentially achieving lossless compression at INT8 precision while enabling deployment on ARM-based mobile processors with 4× reduced memory bandwidth requirements [3]. Advanced calibration methods utilizing exponential moving averages for batch normalization folding further improve quantized model stability, particularly for depthwise separable convolutions, where channel-wise variance significantly impacts quantization error propagation.

Quantization-aware training (QAT) extends these principles by incorporating quantization operations directly into the training process, allowing the network to adapt its learned representations to accommodate reduced numerical precision. This approach proves essential when pursuing aggressive quantization schemes beyond INT8, including binary representations that approach theoretical limits of weight encoding. The XNOR-Net architecture demonstrates extreme binary quantization by constraining both weights and inputs to +1 or -1 values, achieving 58× memory savings and 58× computational

10.48047/jocaaa.2025.34.12.30

acceleration on AlexNet compared to full-precision implementations [4]. On ImageNet classification, XNOR-Net achieves 44.2% top-1 accuracy with binary weights and activations, compared to 56.6% for the full-precision baseline, representing a 12.4 percentage point degradation while enabling convolutional operations to be implemented purely through XNOR bitwise operations followed by bit-counting [4]. The binary weight network variant, which binarizes only weights while maintaining full-precision activations, achieves 56.8% top-1 accuracy on AlexNet, nearly matching full-precision performance while still providing 32× memory compression and eliminating all multiplication operations during inference [4]. Furthermore, on ImageNet classification using ResNet-18 architecture, binary neural networks achieve 51.2% top-1 accuracy compared to 69.3% for the full-precision baseline, delivering practical inference speeds of 58× faster on CPU implementations through replacement of floating-point multiply-accumulate operations with efficient bitwise operations [4].

Dynamic quantization introduces runtime adaptability, adjusting numerical precision based on instantaneous system conditions, including available computational resources, power budget, and thermal state. Implementation requires a lightweight monitoring infrastructure that imposes minimal overhead while providing sufficient telemetry for intelligent precision selection decisions.

Quantization Method	Precision Level	Accuracy Retention	Hardware Benefit	Deployment Context
Post-Training Quantization	INT8	High	Eliminates Floating-Point Hardware	ARM Mobile Processors
Per-Channel Quantization	INT8	Lossless	Reduced Memory Bandwidth	Depthwise Separable Convolutions
Binary Weight Networks	1-bit Weights	Near-Baseline	Multiplication Elimination	Microcontroller Deployment
XNOR-Net	1-bit Full	Moderate	Bitwise Operations Only	Extreme Resource Constraints

Table 2: Quantization Techniques for Model Compression [3, 4]

3. Structured and Unstructured Pruning Techniques

Pruning addresses model compression through selective removal of network parameters, exploiting the overparameterization inherent in modern neural architectures to eliminate redundant computational pathways. The distinction between structured and unstructured pruning fundamentally determines the practical hardware efficiency gains achievable beyond raw parameter count reduction.

Magnitude-based pruning provides an intuitive foundation for parameter elimination, removing weights below a specified threshold under the hypothesis that small-magnitude parameters contribute minimally to network output. Unstructured magnitude pruning examines individual weights in isolation, typically achieving 80-90% sparsity while maintaining acceptable accuracy degradation. The deep compression framework demonstrates that AlexNet can be compressed from 240MB to 6.9MB, achieving 35× compression through a three-stage pipeline combining pruning, trained quantization, and Huffman coding [5]. Initial pruning reduces AlexNet from 61 million parameters to 6.7 million parameters, representing

10.48047/jocaaa.2025.34.12.30

9× compression while maintaining 57.2% top-1 accuracy on ImageNet, identical to the original dense network performance [5]. For VGG-16, which originally contains 138 million parameters occupying 552MB of storage, deep compression achieves a reduction to 11 million parameters stored in 11.3MB, delivering a 49× compression ratio with only 0.3% accuracy degradation from the 68.0% baseline [5]. However, unstructured sparsity patterns provide limited hardware acceleration on conventional processors, as random zero locations prevent effective use of dense matrix operations and SIMD vectorization. Structured pruning addresses this limitation by enforcing regular sparsity patterns aligned with hardware execution units, including channel-wise pruning that eliminates entire convolutional filters. The combination of pruning and 8-bit quantization reduces memory requirements dramatically, with convolutional layers achieving an average of 35× compression and fully-connected layers reaching 40× compression across tested architectures [5].

Hardware-aware pruning elevates optimization beyond generic sparsity metrics to explicitly target the execution characteristics of specific processor architectures. The network slimming approach applies L1 regularization on batch normalization scaling factors during training, automatically identifying and removing unimportant channels to produce compact models with structured sparsity [6]. On VGG-16 trained on CIFAR-10, network slimming reduces parameters from 20.04 million to 2.03 million and decreases FLOPs from 398.44 million to 103.14 million, achieving 74.1% sparsity while maintaining 93.80% accuracy compared to the 93.96% baseline, representing only 0.16% accuracy degradation [6]. For deeper architectures, ResNet-164 on CIFAR-10 demonstrates removal of 60% of channels, reducing parameters from 1.73 million to 0.73 million and FLOPs from 254.47 million to 103.14 million, while achieving 94.35% accuracy versus the 94.54% baseline [6]. On ImageNet classification, network slimming applied to VGG-16 reduces the model from 138 million parameters to 64 million parameters, with top-5 error decreasing from 10.88% to 10.21%, demonstrating that structured pruning can simultaneously improve both efficiency and accuracy through elimination of redundant feature representations [6]. The structured nature of channel-level pruning enables direct acceleration on standard hardware without requiring specialized sparse computation libraries, delivering practical inference speedup proportional to the reduction in computational operations.

Pruning Approach	Sparsity Pattern	Compression Method	Hardware Compatibility	Accuracy Impact
Deep Compression	Unstructured	Multi-Stage Pipeline	Requires Sparse Libraries	Negligible
Network Slimming	Structured Channels	L1 Regularization	Standard Hardware	Minimal Degradation
Magnitude-Based	Individual Weights	Threshold Selection	Limited Acceleration	High Sparsity Achievable
Channel Pruning	Regular Patterns	Batch Normalization Scaling	SIMD-Friendly	Sometimes Improved

Table 3: Pruning Strategies for Neural Network Compression [5, 6]

4. Platform-Specific Implementation and Performance Analysis

10.48047/jocaaa.2025.34.12.30

Validation of optimization techniques requires rigorous evaluation across representative embedded hardware platforms operating under realistic deployment constraints. This section presents empirical results from comprehensive benchmarking campaigns conducted on ARM Cortex-A processors and Raspberry Pi systems, establishing achievable performance envelopes for production edge AI applications.

ARM Cortex-A processors, widely deployed in automotive ECUs and premium mobile devices, provide a representative evaluation platform for high-performance embedded AI. Advanced quantization methodologies demonstrate that 8-bit training of neural networks can match full-precision accuracy while enabling substantial computational acceleration on embedded processors. The scalable 8-bit quantization approach achieves training of ResNet-50 on ImageNet with top-1 accuracy of 76.6%, matching the 76.8% baseline FP32 performance while reducing memory requirements by 75% [7]. This technique employs range batch-normalization that tracks activation statistics during training to determine optimal quantization ranges, preventing accuracy degradation that typically occurs with naive 8-bit conversion [7]. For AlexNet architecture, 8-bit quantized training achieves 57.4% top-1 accuracy compared to 58.8% for the FP32 baseline, representing only 1.4 percentage point degradation while enabling 4× memory compression [7]. The quantization scheme maintains separate precision for different tensor types, with gradients computed at higher precision to preserve training stability while weights and activations utilize 8-bit representation during forward propagation [7]. Critically, this optimization enables real-time inference within the thermal envelope of automotive ECU enclosures, which typically lack active cooling and must maintain stable operation across extreme temperature ranges. The 8-bit quantized networks demonstrate numerical stability across varying computational environments, with consistent accuracy maintained regardless of temperature variations that affect floating-point arithmetic precision in embedded processors [7].

Raspberry Pi platforms and embedded ARM processors represent the lower bound of viable edge AI deployment, requiring collaborative inference strategies to overcome individual device limitations. The Musical Chair framework demonstrates that distributed inference across multiple IoT devices achieves real-time recognition performance unattainable on single embedded platforms [8]. Benchmarking on Raspberry Pi 3 devices shows that individual boards achieve only 0.47 frames per second for VGG-16 inference, insufficient for real-time applications requiring a minimum 10 FPS throughput [8]. However, collaborative execution across four Raspberry Pi devices achieves 1.83 FPS through layer-wise parallelization, representing 3.9× speedup over single-device execution [8]. The framework partitions neural networks across available devices based on computational capabilities and network bandwidth, with ResNet-50 distributed across three collaborative devices, achieving 1.2 FPS compared to 0.31 FPS on isolated execution [8]. Energy efficiency analysis reveals that collaborative inference reduces per-frame energy consumption from 156.3 joules on a single Raspberry Pi to 43.7 joules when distributed across four devices, achieving 3.6× energy efficiency improvement through workload balancing and reduced per-device thermal stress [8]. For MobileNet architectures optimized for embedded deployment, collaborative execution achieves 5.2 FPS across three Raspberry Pi devices with 89.4% parallel efficiency, demonstrating near-linear scaling for communication-efficient network partitioning strategies [8]. Furthermore, the demonstrated capability to dynamically reassign computational workloads based on device availability enables fault-tolerant edge AI systems that maintain operation despite individual device failures in multi-device deployments.

10.48047/jocaaa.2025.34.12.30

Deployment Strategy	Hardware Platform	Optimization Applied	Execution Characteristic	Energy Efficiency
8-bit Training	ARM Cortex-A Processors	Scalable Quantization	Temperature-Stable	Dramatically Reduced
Single Device	Raspberry Pi	Standard Inference	Insufficient for Real-Time	High per Frame
Collaborative Inference	Multiple IoT Devices	Distributed Computation	Near-Linear Scaling	Workload Balancing
Range Batch-Normalization	Automotive ECUs	Adaptive Precision	Thermal Envelope Compliant	Low Power Draw

Table 4: Platform-Specific Embedded AI Performance [7, 8]

5. Practical Implications for Embedded AI Development

The empirical results presented establish several critical implications for the trajectory of edge AI system development and the economic accessibility of intelligent embedded applications.

Hardware accessibility undergoes a fundamental transformation when model resource requirements decrease by an order of magnitude. Applications previously requiring expensive GPU-equipped computing platforms become deployable on low-cost embedded systems, democratizing AI capabilities for educational institutions, developing regions, and price-sensitive consumer applications. The MobileNets architecture family demonstrates that depthwise separable convolutions reduce computational cost dramatically while maintaining competitive accuracy, with MobileNet-1.0 achieving 70.6% ImageNet top-1 accuracy using only 569 million multiply-add operations compared to VGG-16's 15.3 billion operations for 71.5% accuracy [9]. Through width multiplier hyperparameters, MobileNet enables systematic resource-accuracy tradeoffs, with the 0.25 width variant requiring merely 41 million MADDs and achieving 50.6% top-1 accuracy, suitable for deployment on severely resource-constrained microcontrollers [9]. Resolution multipliers provide orthogonal optimization dimensions, where reducing input resolution from 224×224 to 128×128 decreases MobileNet-1.0 computational requirements from 569 million to 138 million MADDs while maintaining 59.1% accuracy, enabling real-time inference on embedded processors operating below 1GHz [9]. This cost reduction extends throughout the product lifecycle, as lower-power processors reduce power supply and thermal management requirements, shrinking both bill-of-materials costs and physical product dimensions. Latency measurements on ARM Cortex-A53 processors demonstrate that MobileNet-0.5 achieves 26ms inference time compared to 900ms for VGG-16, representing 34× speedup while consuming 4.3× fewer parameters [9]. For automotive applications operating under extreme cost optimization pressure, MobileNet architectures enable deployment of vision-based safety features on existing ECU hardware rather than requiring dedicated AI accelerators, with the 0.5 width multiplier variant delivering sufficient accuracy for lane detection and object recognition while executing within 50ms real-time constraints on automotive-grade ARM processors [9].

Energy efficiency improvements manifest as multiplicative benefits across the deployment scale spectrum. Individual edge devices achieve substantial improvements in inference operations per watt through efficient neural architecture design and aggressive model compression. The SqueezeNet architecture achieves AlexNet-level ImageNet accuracy of 57.5% top-1 and 80.3% top-5 while utilizing

10.48047/jocaaa.2025.34.12.30

only 1.25 million parameters, representing 50× parameter reduction compared to AlexNet's 61 million parameters [10]. When compressed using deep compression techniques, SqueezeNet reduces from 4.8MB uncompressed to merely 0.47MB, achieving 510× size reduction relative to AlexNet while maintaining equivalent accuracy [10]. The Fire module design, consisting of squeeze layers with 1×1 convolutions followed by expand layers mixing 1×1 and 3×3 filters, achieves this efficiency through strategic dimensionality reduction that preserves representational capacity while minimizing parameter count [10]. When aggregated across deployments numbering thousands or millions of devices, these efficiency gains yield substantial operational cost reductions. For battery-powered IoT sensors deployed in remote locations, the difference between 4.8MB and 0.47MB model sizes enables deployment on microcontrollers with 8MB flash memory rather than requiring 64MB storage, reducing hardware costs by approximately $5-10 per unit and fundamentally altering deployment economics by reducing maintenance visit frequency through extended battery life enabled by reduced memory access energy consumption [10].

Conclusion

The systematic application of quantization and pruning methodologies enables practical deployment of sophisticated neural network models on embedded systems operating under severe resource constraints. Quantization techniques reduce numerical precision from floating-point to integer representations, achieving memory compression ratios of 4× to 58× while maintaining accuracy degradation below acceptable thresholds for production applications. Post-training quantization provides accessible model compression without retraining requirements, while quantization-aware training recovers accuracy losses inherent in aggressive bitwidth reduction to binary or ternary weight encodings. Pruning approaches exploit the overparameterization characteristic of modern neural architectures, removing redundant parameters through magnitude-based selection or structured channel elimination that aligns with hardware execution characteristics. Deep compression frameworks combining pruning, quantization, and entropy coding achieve compression ratios exceeding 49× on standard architectures while preserving functional accuracy. Hardware-aware optimization strategies explicitly target processor architectures, enforcing sparsity patterns that maximize utilization of SIMD vector units and minimize memory bandwidth consumption that dominates energy budgets in embedded inference workloads. Empirical validation across ARM Cortex-A processors demonstrates that 8-bit quantization combined with structured pruning enables real-time inference within automotive thermal envelopes, while collaborative inference strategies across multiple Raspberry Pi devices overcome individual platform limitations through distributed computation. Efficient neural architectures employing depthwise separable convolutions and compact Fire modules reduce computational requirements by orders of magnitude compared to conventional designs, with MobileNet variants achieving competitive accuracy using merely 569 million multiply-add operations compared to billions required by traditional architectures. The convergence of algorithmic compression techniques and efficient architectural design democratizes artificial intelligence capabilities for deployment on cost-effective embedded platforms, enabling intelligent features in automotive safety systems, battery-powered IoT sensors, and consumer electronics operating under extreme cost optimization pressure. These advancements fundamentally transform the economic accessibility of edge artificial intelligence, reducing hardware costs by enabling deployment on processors costing tens rather than hundreds of dollars while extending battery life through dramatic reductions in memory access energy consumption. The demonstrated feasibility of deploying complex vision and language models on resource-constrained platforms opens new application domains in remote sensing, industrial automation,

10.48047/jocaaa.2025.34.12.30

and mobile robotics, where latency, privacy, or connectivity constraints preclude cloud-based inference architectures.

References

[1] Yiwen Guo, et al., "Dynamic Network Surgery for Efficient DNNs," arxiv, 2016. Available: https://arxiv.org/abs/1608.04493

[2] Nicholas D. Lane, et al., "DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices," IEEE, 2016, Available: https://ieeexplore.ieee.org/document/7460664

[3] Raghuraman Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," ResearchGate, 2018. Available:https://www.researchgate.net/publication/325922339_Quantizing_deep_convolutional_networks_for_efficient_inference_A_whitepaper

[4] Mohammad Rastegari, et al., "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," arxiv, 2016. Available: https://arxiv.org/abs/1603.05279

[5] Song Han, et al., "DEEP COMPRESSION: COMPRESSING DEEP NEURAL NETWORKS WITH PRUNING, TRAINED QUANTIZATION AND HUFFMAN CODING," UW Homepage, 2016. Available: https://courses.cs.washington.edu/courses/cse550/22au/papers/CSE550.DeepCompression.pdf

[6] Zhuang Liu, et al., "Learning Efficient Convolutional Networks through Network Slimming," IEEE, 2017, Available: https://arxiv.org/abs/1708.06519

[7] Ron Banner, et al., "Scalable Methods for 8-bit Training of Neural Networks," arxiv, 2018. Available: https://arxiv.org/abs/1805.11046

[8] Ramyad Hadidi, et al., "Real-Time Image Recognition Using Collaborative IoT Devices," ACM Digital Library, 2018.. Available: https://dl.acm.org/doi/10.1145/3229762.3229765

[9] Andrew G. Howard, et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv, 2017. Available: https://arxiv.org/abs/1704.04861

[10] Forrest N. Iandola, et al., "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," ResearchGate, 2016. Available: https://www.researchgate.net/publication/301878495_SqueezeNet_AlexNet-level_accuracy_with_50x_fewer_parameters_and_05MB_model_size