Universal Library of Engineering Technology Research Article

ISSN: 3064-996X | Volume 3, Issue 1

Open Access | PP: 07-11

DOI: https://doi.org/10.70315/uloap.ulete.2026.0301002

# Edge AI for Real-Time Robotic Systems: Architectures, Deployment Strategies, and Performance Optimization

**Ashis Ghosh**

Independent Researcher, CA, USA.

## Abstract

*The deployment of artificial intelligence on edge devices has emerged as a critical enabler for real-time robotic systems, where latency constraints and computational efficiency directly impact system performance and safety. This paper presents a comprehensive analysis of edge AI architectures and deployment strategies for robotic applications, examining the interplay between hardware platforms, model optimization techniques, and application-specific requirements. We systematically evaluate the performance characteristics of modern edge computing platforms, including the NVIDIA Jetson family, and assess optimization techniques such as quantization, pruning, and inference acceleration frameworks. Drawing from established patterns in modular robotic architectures and closed-loop control systems, we propose a five-stage deployment methodology that guides practitioners through requirement analysis, model selection, optimization, hardware alignment, and system integration. Experimental evaluation demonstrates that optimized edge deployments achieve inference latencies below 10 milliseconds—an order of magnitude improvement over cloud-based processing—while maintaining accuracy within 2% of full-precision baselines. The findings provide actionable guidance for robotics practitioners seeking to deploy AI capabilities on resource-constrained embedded platforms while meeting stringent real-time requirements.*

**Keywords:** *Artificial Intelligence, Edge Computing, Embedded Systems, Real-Time Systems, Robotic Manipulation.*

## INTRODUCTION

The integration of artificial intelligence into robotic systems has transformed capabilities across domains ranging from industrial automation to service robotics. However, deploying AI models in real-time robotic applications presents fundamental challenges that distinguish these systems from cloud-based AI services. Robotic systems operating in dynamic environments require rapid perception-to-action cycles, where delays of even tens of milliseconds can compromise safety, efficiency, and task performance [1].

Traditional approaches that offload AI inference to cloud servers introduce network latency that proves incompatible with real-time control requirements. Cloud-based processing typically incurs round-trip latencies of 100 milliseconds or more, whereas autonomous vehicle perception systems require end-to-end processing within 10 milliseconds for safe operation at highway speeds [2]. This latency gap has driven the emergence of edge AI—the deployment of AI models directly on embedded devices co-located with sensors and actuators.

Modern robotic systems increasingly adopt modular software architectures that separate perception, reasoning, and control into distinct layers with well-defined timing contracts [3]. This architectural pattern enables the deployment of computationally intensive AI models at the perception layer while maintaining deterministic timing for safety-critical control loops. The closed-loop control strategies employed in robotic manipulation, where grasp parameters are continuously updated based on visual feedback [4], exemplify applications where edge AI deployment is essential for achieving the required response times.

The edge AI ecosystem has matured significantly, with hardware platforms now offering hundreds of tera-operations per second (TOPS) within power envelopes suitable for mobile and embedded deployment [5]. Simultaneously, software frameworks and optimization techniques have advanced to enable efficient deployment of complex neural networks on resource-constrained devices [6]. However, effectively leveraging these capabilities requires systematic approaches that address the unique constraints of robotic applications.

This paper presents a comprehensive analysis of edge AI for real-time robotic systems. The contributions include: (1) a systematic evaluation of edge computing platforms and their

**Citation:** Ashis Ghosh, "Edge AI for Real-Time Robotic Systems: Architectures, Deployment Strategies, and Performance Optimization", Universal Library of Engineering Technology, 2026; 3(1): 07-11. DOI: https://doi.org/10.70315/uloap.ulete.2026.0301002.

suitability for robotic applications; (2) an assessment of model optimization techniques and their impact on inference performance; (3) a five-stage deployment methodology derived from current best practices; and (4) experimental validation demonstrating achievable performance characteristics for representative robotic workloads.

The remainder of this paper is organized as follows: Section 2 presents the materials and methods, including hardware platforms, optimization techniques, and evaluation methodology. Section 3 presents results and discussion covering performance benchmarks, optimization trade-offs, and integration considerations. Section 4 concludes with practical recommendations for practitioners.

## MATERIALS AND METHODS

### Edge Computing Platforms

The evaluation encompasses representative edge computing platforms spanning the performance-power spectrum relevant to robotic applications.

NVIDIA Jetson Family: The Jetson platform provides GPU-accelerated inference optimized for robotics and autonomous systems. The Jetson AGX Orin delivers up to 275 TOPS of AI performance with power configurable between 15W and 60W, representing an 8X performance improvement over the previous-generation Jetson AGX Xavier [5]. The recently announced Jetson Thor further advances capabilities to 2,070 FP4 TFLOPS within a 130W envelope, providing 7.5X higher AI compute than the AGX Orin [7].

Google Coral TPU: The Coral Edge TPU provides 4 TOPS of inference performance at 2W power consumption, suitable for lower-complexity models in power-constrained applications [8].

Qualcomm Robotics Platforms: The RB5 and RB6 platforms combine heterogeneous computing with 5G connectivity, targeting autonomous mobile robots and drones with 15 TOPS AI performance [9].

### Model Optimization Techniques

Deploying neural networks on edge devices requires optimization techniques that reduce computational requirements while preserving model accuracy.

Quantization: Reducing numerical precision from 32-bit floating point (FP32) to lower bit-widths significantly accelerates inference. INT8 quantization typically achieves 3-4X speedup compared to FP32, with accuracy loss under 1-2% for well-calibrated models [6]. Post-training quantization (PTQ) offers simplicity without retraining, while quantization-aware training (QAT) produces superior accuracy by simulating quantization during training [10].

Pruning: Removing redundant weights and neurons reduces model size and computational requirements. Structured pruning removes entire filters or channels, enabling direct acceleration on standard hardware, while unstructured pruning achieves higher compression ratios but requires specialized sparse computation support [11].

Knowledge Distillation: Training compact student networks to mimic larger teacher networks transfers capabilities to deployment-efficient architectures [12]. Neural Architecture Search (NAS) automates discovery of efficient architectures optimized for specific hardware targets [13].

### Inference Acceleration Frameworks

Software frameworks optimize neural network execution for target hardware. TensorRT, NVIDIA's inference optimizer, applies layer fusion, kernel auto-tuning, precision calibration, and memory optimization to maximize throughput on GPU hardware [6]. ONNX Runtime provides cross-platform inference with execution providers for various accelerators [14]. TensorFlow Lite supports mobile and embedded deployment with quantization and hardware delegation [15].

### Evaluation Methodology

Performance evaluation employed standardized workloads representative of robotic perception tasks: object detection (YOLOv5, YOLOv8), pose estimation (MediaPipe Pose, MoveNet), semantic segmentation (DeepLabV3, BiSeNet), and grasp assessment networks for manipulation planning informed by visual servoing requirements [4].

Metrics captured include inference latency (p50, p95, p99), throughput (FPS), accuracy (mAP, PCK), power consumption, and memory utilization. All experiments were conducted on NVIDIA Jetson AGX Orin configured at 60W power mode with JetPack 6.0, TensorRT 10.0, and CUDA 12.2.

## RESULTS AND DISCUSSION

### Platform Performance Benchmarks

Table 1 presents inference latency results for object detection models across optimization configurations on the Jetson AGX Orin platform.

**Table 1.** Object Detection Inference Latency (Jetson AGX Orin)

| Model | Precision | Latency (ms) | Throughput (FPS) | mAP |
|---|---|---|---|---|
| YOLOv5n | FP32 | 8.2 | 122 | 28.0 |
| YOLOv5n | FP16 | 4.1 | 244 | 27.9 |
| YOLOv5n | INT8 | 2.8 | 357 | 27.4 |
| YOLOv5s | FP32 | 12.4 | 81 | 37.4 |
| YOLOv5s | INT8 | 4.1 | 244 | 36.8 |
| YOLOv8n | INT8 | 2.6 | 385 | 36.7 |

INT8 quantization achieves 2.9-3.0X speedup compared to FP32 while retaining 98.4% of baseline accuracy, consistent with reported findings that quantization delivers significant acceleration with minimal accuracy degradation [6]. The YOLOv8n model with INT8 precision achieves sub-3ms inference latency, enabling perception rates exceeding 300Hz.

## Comparison with Cloud Processing

Table 2 contrasts edge deployment latency against cloud-based inference, demonstrating the order-of-magnitude improvement achieved through edge processing.

**Table 2.** Edge vs. Cloud Inference Latency Comparison

| Processing Location | Network (ms) | Inference (ms) | Total (ms) |
|---|---|---|---|
| Cloud (GPU Server) | 80-120 | 5-10 | 85-130 |
| Edge (Jetson Orin) | 0 | 3-8 | 3-8 |
| Edge (Coral TPU) | 0 | 8-15 | 8-15 |

Edge deployment eliminates network latency entirely, achieving total latencies under 10ms compared to 100ms+ for cloud processing. This improvement is critical for robotic applications where research indicates that teleoperation latency below 170ms has minor impact on operator performance, while latency exceeding 700ms makes real-time interaction nearly impossible [16]. Visual servoing systems for robotic manipulation typically require perception latencies below 33ms to support 30Hz control loops [4].

## Optimization Technique Analysis

Table 3 summarizes the impact of individual optimization techniques on the YOLOv5s model.

**Table 3.** Optimization Technique Impact on YOLOv5s

| Technique | Speedup | Accuracy Retention | Size Reduction |
|---|---|---|---|
| FP16 Conversion | 2.0X | 99.8% | 50% |
| INT8 PTQ | 3.0X | 98.4% | 75% |
| INT8 QAT | 3.0X | 99.2% | 75% |
| Pruning (50%) | 1.8X | 98.0% | 50% |
| Pruning + INT8 | 4.2X | 97.1% | 87% |

Quantization-aware training (QAT) preserves 99.2% of baseline accuracy while achieving 3.0X speedup, outperforming post-training quantization. Combined pruning and quantization achieves 4.2X speedup with 97.1% accuracy retention. The results align with comprehensive surveys reporting 2-4X model compression and 20-80% computational savings through quantization techniques [17].

## Five-Stage Deployment Methodology

Based on patterns identified across successful edge AI deployments and established robotic system design principles [3], we propose a systematic five-stage methodology: (1) Requirement Definition—establishing quantitative targets for latency, accuracy, power, and memory; (2) Model Selection—choosing architectures from the accuracy-latency Pareto frontier; (3) Optimization—applying quantization, pruning, and architecture modifications; (4) Hardware Alignment—configuring inference frameworks for target hardware; and (5) System Integration—integrating optimized models with appropriate scheduling and resource management.

## Robotic System Integration

Effective edge AI deployment requires integration with broader robotic system architectures. The modular software architecture for mobile manipulation systems described by Ghosh [3] provides a template where AI perception components operate within timing contracts that preserve control loop determinism.

For manipulation tasks employing closed-loop grasping with visual servoing [4], edge AI enables the continuous perception updates required for feedback control. Table 4 demonstrates achieved performance for a complete visual servoing pipeline.

**Table 4.** Visual Servoing Pipeline Performance

| Component | Latency (ms) | Target (ms) |
|---|---|---|
| Image Capture | 2.1 | <5 |
| Object Detection | 4.1 | <10 |
| Pose Estimation | 6.2 | <15 |
| Grasp Planning | 3.4 | <10 |
| Total Pipeline | 15.8 | <33 |

The total pipeline latency of 15.8ms supports 60Hz perception updates, exceeding the 30Hz minimum for stable visual servoing control. This performance level enables the continuous grasp parameter recalculation during arm motion described in closed-loop grasping techniques [4].

## Power and Thermal Considerations

Edge deployment for mobile robots requires careful power management. The Jetson AGX Orin's configurable power modes enable dynamic adaptation based on battery state and thermal conditions. At 30W—half the maximum power—the system still achieves 147 FPS for YOLOv5s INT8 inference, sufficient for most robotic perception requirements while extending operational duration for mobile platforms.

## Discussion

The experimental results validate edge AI as a mature technology for real-time robotic systems. Key findings include: sub-10ms inference latency achievable for object detection and pose estimation; quantization-aware training preserving 99% of baseline accuracy while achieving 3X speedup; the five-stage deployment methodology complementing established robotic software architecture patterns [3]; and edge AI performance supporting visual servoing control loops at 30-60Hz, enabling closed-loop manipulation strategies [4].

Limitations include thermal management for sustained operation, model-specific optimization requirements, and the need for application-specific accuracy validation. Future edge platforms including Jetson Thor promise further performance improvements [7].

## CONCLUSION

This paper presented a comprehensive analysis of edge AI deployment for real-time robotic systems. Through systematic evaluation of hardware platforms, optimization techniques, and integration strategies, we demonstrated that current edge computing technology enables AI inference at latencies compatible with robotic control requirements.

Key findings include: (1) INT8 quantization achieves 3X speedup with under 2% accuracy loss; (2) edge deployment reduces inference latency by 10X compared to cloud processing; (3) visual servoing pipelines achieve total latencies under 20ms, supporting 30-60Hz control loops; and (4) the five-stage deployment methodology provides actionable guidance for practitioners.

For roboticists deploying AI capabilities on embedded platforms, we recommend: prioritizing quantization-aware training when accuracy margins permit; maintaining model portfolios for runtime adaptation; integrating AI components within modular architectures with explicit timing contracts; and validating end-to-end system performance under realistic operating conditions.

Future work will explore dynamic model selection based on computational headroom, integration with emerging hardware platforms, and automated optimization pipelines that adapt to evolving model architectures.

## REFERENCES

1. Sculley D, Holt G, Golovin D, et al. Hidden technical debt in machine learning systems. Advances in Neural Information Processing Systems. 2015;28:2503-2511.
2. Liu S, Liu L, Tang J, Yu B, Wang Y, Shi W. Edge computing for autonomous driving: Opportunities and challenges. Proceedings of the IEEE. 2019;107(8):1697-1716.
3. Ghosh A. A modular software architecture for safe and scalable mobile manipulation systems. International Journal of Engineering Technology and Computer Science IT Innovations. In press.
4. Augenbraun JE, Ghosh A, Hansen SJ, Verheye A, MacPhee D. Robot for performing dextrous tasks and related methods and systems. U.S. Patent 11,407,118 B1. August 9, 2022.
5. NVIDIA Corporation. Jetson AGX Orin for next-generation robotics. NVIDIA Developer. 2024.
6. NVIDIA Corporation. TensorRT SDK. NVIDIA Developer. 2024.
7. NVIDIA Corporation. Introducing NVIDIA Jetson Thor, the ultimate platform for physical AI. NVIDIA Technical Blog. 2025.
8. Google. Coral Edge TPU. Google Coral. 2024.
9. Qualcomm Technologies. Qualcomm Robotics RB6 Platform. Qualcomm. 2024.
10. NVIDIA Corporation. Achieving FP32 accuracy for INT8 inference using quantization aware training with TensorRT. NVIDIA Technical Blog. 2021.
11. Cheng Y, Wang D, Zhou P, Zhang T. A survey of model

compression and acceleration for deep neural networks. IEEE Signal Processing Magazine. 2018;35(1):126-136.

12. Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531. 2015.
13. Zoph B, Le QV. Neural architecture search with reinforcement learning. In: Proc. ICLR; 2017.
14. Microsoft. ONNX Runtime. Microsoft Open Source. 2024.
15. Google. TensorFlow Lite. TensorFlow. 2024.
16. Schimpe A, Betz J, Lienkamp M. Network latency in teleoperation of connected and autonomous vehicles: A review of trends, challenges, and mitigation strategies. Electronics. 2024;13(12):2224.
17. Bakhtiarnia M, Zhang Q, Iosifidis A. Early-exit deep neural networks: A comprehensive survey. ACM Computing Surveys. 2024;57(6):1-35.
18. Kouris A, Venieris SI, Bouganis CS. Edge AI in practice: A survey and deployment framework for neural networks on embedded systems. Electronics. 2024;14(24):4877.
19. ACM Computing Surveys. Empowering edge intelligence: A comprehensive survey on on-device AI models. ACM Computing Surveys. 2024.
20. Edge AI and Vision Alliance. Optimizing edge AI for effective real-time decision making in robotics. 2025.