International Journal for Research Publication and Seminar

ISSN: 2278-6848 | Vol. 16 | Issue 1 | Jan-Mar 2025 | Peer Reviewed & Refereed

Special Edition : SPARK 2025 : XXI National Conference on Emerging Technology Trends in Engineering & Project Competition

A Review of TinyML-Based Object Detection for Autonomous Robotic Arms: Challenges and Comparisons with Traditional Methods

Minal Thawakar[1], Pranav Shende[2], Arun Kayshap[3], Sujal Borkar[4], Krishanam Bilse[5]

Artificial Intelligence and Data Science, KDK College of Engineering Nagpur, India

[1]*minalthawakar1988@gmail.com*

[2]pranav779895@gmail.com,

[3]golukashyap7517@gmail.com,

[4]borkarsujal13@gmail.com,

[5]krishnambilse122@gmail.com

***Abstract*—** Tiny Machine Learning (TinyML) is a rapidly growing field at the intersection of artificial intelligence and embedded systems, enabling machine learning (ML) applications on low-power, resource-constrained devices such as microcontrollers (MCUs). The adoption of TinyML in autonomous robotics, particularly for pick-and-place tasks, has the potential to revolutionize warehouse automation. This review paper explores the methodologies, tools, applications, and challenges of TinyML-based object detection in autonomous robotic arms and compares it with traditional methods relying on cloud computing and high-power GPUs. We discuss key advancements in TinyML frameworks and inference engines, highlighting the trade-offs in accuracy, power consumption, and real-time performance. Additionally, we analyze the constraints associated with deploying TinyML models on microcontrollers such as the ESP32-CAM and compare their efficiency against traditional machine learning approaches for robotic arms. Finally, we present potential future research directions in TinyML for robotic applications.

***Keywords*—**TinyML, Robotic Arm, Object Detection, Edge AI, On-Device Learning, Warehouse Automation.

INTRODUCTION

The field of Artificial Intelligence (AI) and Machine Learning (ML) has experienced significant advancements, leading to widespread adoption in various industries, including robotics and automation. Traditionally, ML models are deployed on powerful computing infrastructures such as cloud servers, edge devices with GPUs/TPUs, and high-performance computing clusters. These models typically require large computational power, high memory (often in gigabytes), and substantial energy consumption, making them infeasible for deployment on low-power, resource-constrained microcontrollers (MCUs). The increasing demand for real-time, on-device intelligence in autonomous

systems has led to the emergence of Tiny Machine Learning (TinyML), a paradigm shift that enables ML inference on low-power microcontrollers. By running ML models on devices like the ESP32-CAM and STM32, TinyML eliminates cloud dependency, reducing latency, power consumption, and data transmission costs.

One of the most promising applications of TinyML is in robotic automation, particularly warehouse logistics, where autonomous robotic arms perform pick-and-place tasks. Traditionally, object detection and decision-making in robotic systems rely on computationally expensive models such as Convolutional Neural Networks (CNNs) running on GPUs, TPUs, or cloud servers. While these models provide high accuracy, they demand significant computational power, increasing operational costs and energy consumption. Furthermore, cloud-based ML solutions suffer from latency, privacy concerns, and connectivity dependence, making them less suitable for real-time robotic control.

TinyML provides a promising alternative by enabling real-time, low-power object detection directly on microcontrollers like the ESP32-CAM. It utilizes model quantization, neural architecture search (TinyNAS), pruning, and lightweight frameworks (TensorFlow Lite Micro, MCUNet, TinyTL) to efficiently deploy deep learning models on microcontrollers with extremely limited resources (typically <256KB RAM and 1mW power consumption). This makes TinyML particularly attractive for autonomous robotic arms, where fast, low-energy, on-device decision-making is essential for efficient and scalable warehouse logistics automation. The objective of this review paper is to analyze the current advancements, tools, and applications of TinyML for object detection in robotic arms. Furthermore, the paper provides a detailed comparison between TinyML-based and traditional ML-based approaches, highlighting trade-offs in terms of accuracy, power consumption, latency, and computational efficiency. Finally, the paper discusses key challenges and future directions in deploying TinyML-powered robotic systems, with a focus on improving model efficiency, reducing inference time, and optimizing energy consumption for real-world applications.

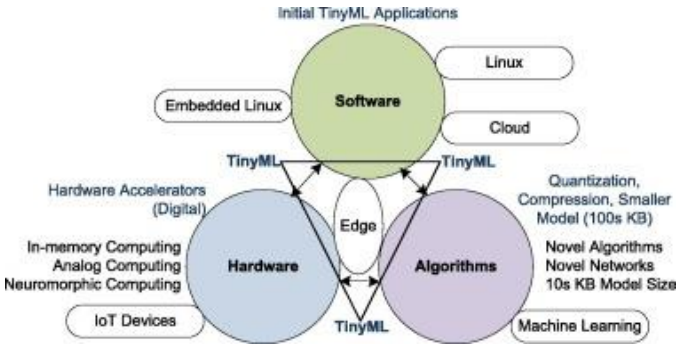

Fig1: Tiny ML representation

II. TRADITIONAL ML AND TINY ML IN ROBOTIC AUTOMATION

Traditional ML and deep learning techniques have been widely used in autonomous robotics, including object detection, path planning, and decision-making. These approaches often rely on powerful hardware such as GPUs (e.g., NVIDIA Jetson), TPUs, and cloud-based AI servers, which can handle computationally expensive models like convolutional neural networks (CNNs) and deep reinforcement learning algorithms. These methods typically require significant memory, high power consumption, and stable network connectivity for offloading computations to the cloud.

In contrast, TinyML offers an alternative approach by enabling ML models to run locally on microcontrollers (e.g., ESP32-CAM, STM32, ARM Cortex-M) with extremely low power consumption (typically <1mW). This paradigm shift eliminates cloud dependency, reducing latency, increasing data privacy, and making robotic systems more autonomous. In warehouse robotics, TinyML-based object detection allows a robotic arm to identify and classify objects in real-time while operating on low-power embedded devices. However, compared to traditional ML approaches, TinyML faces challenges such as lower accuracy, limited model complexity, and slower inference speed due to the constraints of microcontroller hardware.

A key comparison between TinyML-based object detection and traditional ML-based robotic automation is necessary to evaluate the advantages and trade-offs of each approach. Traditional ML models, such as YOLO, MobileNet, and SSD, demand high memory (>1GB RAM) and computational power (multi-core CPUs or GPUs running at GHz speeds). These models offer higher accuracy but require internet connectivity to communicate with cloud-based servers, leading to latency, security risks, and high energy consumption. This makes them less practical for real-time, battery-operated robotic systems.

In comparison, TinyML-based object detection models are optimized using techniques such as quantization (e.g., INT8, INT4), TinyNAS (neural architecture search), and MCUNet (model optimization for MCUs). These allow object detection models to run locally on devices like ESP32-CAM, STM32, and Raspberry Pi Pico, significantly reducing power consumption. However, this compression often leads to trade-offs in accuracy, affecting object detection performance.

One of the key applications of TinyML in robotics is object detection and control in autonomous systems. Traditional computer vision models, such as YOLO (You Only Look Once), SSD (Single Shot Multibox Detector), and Faster R-CNN, require extensive computational resources, making them impractical for deployment on microcontrollers. Instead, TinyML-based solutions leverage optimized neural

architectures, enabling inference on low-power devices such as ESP32-CAM and STM32.

Recent research has introduced model compression techniques, neural architecture search (NAS), and quantization to adapt ML models for microcontrollers. Several frameworks, such as MCUNet, TinyNAS, TensorFlow Lite Micro (TF-Lite Micro), CMSIS-NN, and TinyTL, have been designed specifically for constrained environments. These frameworks optimize inference speed, power efficiency, and memory usage while maintaining acceptable accuracy levels.

Key advancements in TinyML for robotics include:

1. MCUNet & TinyNAS: Efficient neural architecture search (NAS) techniques that generate models optimized for microcontrollers.
2. TinyTL & TinyOL: Methods for on-device transfer learning and online learning, enabling adaptation to new objects in dynamic environments.
3. Edge Impulse & TF-Lite Micro: Low-power inference frameworks that allow ML models to be deployed on ESP32, STM32, and ARM Cortex-M microcontrollers.
4. Performance comparison of hardware: Studies comparing inference speeds on ESP32-CAM, Raspberry Pi, and Jetson Nano, revealing trade-offs between accuracy, latency, and energy efficiency.

Fig2: The advancement of computing technology over time

While traditional ML methods provide superior accuracy, their reliance on high-performance processors, cloud-based computation, and continuous connectivity makes them impractical for low-power, real-time applications. In contrast, TinyML enables autonomous robotic control with lower power requirements and minimal latency, making it an attractive alternative for warehouse automation, industrial robotics, and remote monitoring applications.

III. CHALLENGES IN TINY ML ROBOTICS

Despite its advantages, TinyML faces several limitations compared to traditional AI in robotics:

1. Memory Constraints: TinyML models operate within 256KB–1MB RAM, whereas traditional AI models require several gigabytes of RAM.
2. Computational Cost: TinyML devices run at 100–500MHz, while GPUs operate at GHz speeds, leading to performance trade-offs.
3. Training on MCUs: Unlike traditional ML, which allows continuous model training, TinyML primarily supports inference, limiting adaptability.
4. Model Compression & Accuracy Trade-offs: TinyML relies on quantization (INT8, INT4) and pruning, often reducing model accuracy.
5. Limited Development Frameworks: While traditional ML benefits from robust frameworks like TensorFlow and PyTorch, TinyML relies on TF-Lite Micro, Edge Impulse, and MCUNet, which are still evolving.

IV. FUTURE RESEARCH DIRECTIONS IN TINY ML FOR ROBOTICS

To enhance TinyML adoption in robotics, future research should focus on several critical areas to address current limitations and unlock new capabilities. One of the most pressing challenges is improving neural architecture search (NAS) and quantization techniques. Developing high-accuracy, ultra-low-power models through advanced NAS approaches will enable TinyML to achieve better performance without sacrificing computational efficiency. Additionally, more refined quantization techniques, such as adaptive mixed-precision quantization, could enhance model compression without significant loss of accuracy.

Another crucial research direction is on-device training. Current TinyML frameworks primarily support inference, but the ability to perform continuous learning and adaptation directly on microcontrollers would greatly expand their usefulness in dynamic environments. Emerging approaches such as TinyTL (Tiny Transfer Learning) and federated learning for microcontrollers offer promising avenues for enabling on-device updates while minimizing power consumption.

Furthermore, hybrid edge-cloud AI systems present an opportunity to balance on-device inference and cloud-assisted learning. While TinyML provides real-time, low-power computation, some tasks may require periodic cloud-based processing for model refinement or heavier computations. Future research should explore intelligent task allocation mechanisms that dynamically distribute workloads between the edge and the cloud to optimize energy efficiency and latency.

Another critical area is energy-efficient hardware accelerators for TinyML. Custom AI hardware designed specifically for low-power microcontrollers—such as dedicated AI co-processors, neuromorphic computing architectures, and RISC-V-based accelerators—can significantly improve inference speed while reducing power consumption. Developing such hardware tailored for TinyML-based robotic applications will be essential for achieving higher efficiency and real-time performance.

Finally, research must focus on expanding TinyML's application domains within robotics. While current implementations are primarily centered on object detection and classification, future advancements should explore motion planning, adaptive control, and multi-modal sensor fusion for more complex robotic interactions. Integrating TinyML with other low-power sensing technologies such as LiDAR, ultrasonic sensors, and radar could further enhance its capabilities in robotic automation.

V. CONCLUSION

TinyML is rapidly transforming robotic automation by enabling real-time, low-power, and efficient object detection on microcontrollers. By eliminating reliance on cloud-based AI and high-performance GPUs, TinyML provides a cost-effective and energy-efficient solution for deploying machine learning in warehouse logistics, industrial robotics, and autonomous pick-and-place systems.

Despite its advantages, TinyML faces several challenges, including memory constraints, model compression trade-offs, and limited support for on-device learning. However, recent advancements in neural architecture search (NAS), quantization, and hybrid edge-cloud AI are helping to mitigate these limitations and improve the feasibility of deploying TinyML-based robotic automation at scale.

Future research should focus on enhancing model efficiency, developing specialized AI hardware accelerators, and expanding Tiny MLs applications in robotics beyond object detection. By addressing these challenges, TinyML will continue to bridge the gap between embedded systems and AI, leading to more intelligent, autonomous, and power-efficient robotic solutions in real-world applications.

As the field of TinyML evolves, it holds the potential to redefine how AI is integrated into low-power robotics, fostering the development of autonomous robotic systems that are faster, smarter, and more sustainable in an increasingly AI-driven world.

REFERENCES

[1] M. Mazumder, C. Banbury, J. Meyer, P. Warden, and V. J. Reddi, "Few-shot keyword spotting in any language," arXiv preprint arXiv:2104.01454, 2021.

[2] E. Hardy and F. Badets, "An ultra-low power rnn classifier for always-on voice wake-up detection robust to real-world scenarios," arXiv preprint arXiv:2103.04792, 2021.

[3] T. Luukkonen, A. Colley, T. Seppänen, and J. Häkkilä, "Cough activated dynamic face visor," in Augmented Humans Conference 2021, 2021, pp. 295–297.

[4] Hsiao-Yun Tseng, Ching-Hao lai, Shyr-Shen Yu, "An effective license plate detection method for over exposure and complex vehicle images," Internationa Conference on Convergence and Hybrid Information Technology , 2008, pp. 176-181.

[5] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, "Towards total recall in industrial anomaly detection," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 14 318–14 328.

[6] A. Capotondi, M. Rusci, M. Fariselli, and L. Benini, "Cmix-nn: Mixed low-precision cnn library for memory constrained edge devices," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 5, pp. 871–875, 2020.

[7] M. Muniswamaiah, T. Agerwala, and C. C. Tappert, "A survey on cloudlets, mobile edge, and fog computing," in 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom). IEEE, Jun 2021, p. 139–142.

[8] S. G. Patil, P. Jain, P. Dutta, I. Stoica, and J. Gonzalez, "Poet: Training neural networks on tiny devices with integrated dematerialization and paging," in International Conference on Machine Learning. PMLR, 2022, pp. 17 573–17 583

[9] C. Profentzas, M. Almgren, and O. Landsiedel, "Minilearn: On-device learning for low-power iot devices," in Proceedings of the 2022 International Conference on Embedded Wireless Systems and Networks (Linz, Austria)(EWSN 22). Junction Publishing, USA, 2022.

[10] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1365– 1374.

[11] D. L. Dutta and S. Bharali, "TinyML meets IoT: A comprehensive

[12] survey," Internet Things, vol. 16, Dec. 2021, Art. no. 100461.

[13] C. Banbury et al., "MLPerf Tiny benchmark," in Proc. Adv. Neural Inf.Process. Syst., 2021, 1–15.

[14] P. Warden, "Speech commands: A dataset for limited-vocabulary
speech recognition," 2018, arXiv:1804.03209.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit.
(CVPR), Jun. 2016, pp. 770–778.

[16] R. Krishnamoorthi, "Quantizing deep convolutional networks for effi-cient inference: A whitepaper," 2018, arXiv:1806.08342.

[17] O. Saha, A. Kusupati, H. V. Simhadri, M. Varma, and P. Jain, "RNNPool: Efficient non-linear pooling for RAM constrained infer-ence," in Proc. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 1–12.

[18] Christos Nikolaos E. Anagnostopoulos, Ioannis E. Anagnostopoulos, Vassili Loumos, and Eleftherios Kayafas, "A License Plate-Recognition Algorithm for Intelligent Transportation System Applications," pp. 377-392, 2006.

[19] Edge Impulse. Accessed: Mar. 3, 2022. [Online]. Available: https://www.edgeimpulse.com/

[20] S. Yao, Y. Zhao, A. Zhang, L. Su, and T. Abdelzaher, "DeepIoT: Compressing deep neural network structures for sensing systems with a compressor-critic framework," in Proc. 15th ACM Conf. Embedded Netw. Sensor Syst., Nov. 2019, pp. 1–14.

[21] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020, pp. 1–16.

[22] H. Cai, C. Gan, L. Zhu, and S. Han, "TinyTL: Reduce memory, not parameters for efficient on-device learning," in Proc. Adv. Neural Inf.Process. Syst., vol. 33, 2020, pp. 1–13.

[23] I. Fedorov, R. P. Adams, M. Mattina, and P. N. Whatmough, "SpArSe:Sparse architecture search for CNNs on resource-constrained micro-controllers," in Proc. Adv. Neural Inf. Process. Syst., vol. 32, 2019,pp. 1–13.

[24] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–21.

[25] J. Yu, A. Lukefahr, D. Palframan, G. Dasika, R. Das, and S. Mahlke, "Scalpel: Customizing DNN pruning to the underlying hardware par-allelism," in Proc. 44th Annu. Int. Symp. Comput. Archit., Jun. 2017,pp. 548–560.

[26] E. Liberis and N. D. Lane, "Differentiable network pruning to enable smart applications," in Proc. 4th U.K. Mobile, Wearable Ubiquitous Syst. Res. Symp., 2022, p. 1.

[27] U. Thakker et al., "Compressing RNNs to kilobyte budget for IoT devices using Kronecker products," ACM J. Emerg. Technol. Comput.Syst., vol. 17, no. 4, pp. 1–18, Jul. 2021.

[28] U. Thakker, P. Whatmough, Z. Liu, M. Mattina, and J. Beu, "Doping: A technique for extreme compression of LSTM models using sparse structured additive matrices," in Proc. Mach. Learn. Syst., vol. 3, 2021, pp. 533–549.

[29] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size," 2016, arXiv:1602.07360.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton "ImageNet classification with deep convolutional neural networks," in Proc. Adv. Neural Inf. Process. Syst. (NIPS), vol. 25, Dec. 2012, pp. 1097–1105.

[31] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen,"MobileNetV2: Inverted residuals and linear bottlenecks," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018,pp. 4510–4520.

[32] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018,pp. 6848–6856.