Embedded Systems for Edge AI: Hardware, Software, and Design Methodologies

Research Proposal · November 2022

DOI: 10.13140/RG.2.2.29378.64962

1 author:

Vasuki Shankar
NVIDIA
14 PUBLICATIONS **125** CITATIONS
SEE PROFILE

Proposed Study for Research Grant

.

Embedded Systems for Edge AI: Hardware, Software, and Design Methodologies

by

Vasuki Shankar[1]

[1]Orcid ID: https://orcid.org/0009-0001-3094-7161

Abstract: The rapid evolution of edge computing has redefined how intelligence is distributed across devices, leading to a paradigm shift from cloud-centric AI to localized, on-device processing. Edge Artificial Intelligence (Edge AI) enables computation closer to the data source, ensuring low latency, enhanced privacy, improved energy efficiency, and real-time decision-making. This paper examines the foundational aspects of embedded Edge AI systems, focusing on hardware architectures, software frameworks, and co-design methodologies that collectively optimize performance under resource-constrained environments. Practical implementations in domains such as healthcare, industrial IoT, robotics, and intelligent vision are explored to demonstrate the transformative impact of Edge AI on modern embedded systems. Key challenges such as task scheduling, secure inference, lifelong learning, and domain-specific hardware design are analyzed in the context of emerging research directions. Finally, the paper highlights ongoing efforts toward standardization, interoperability, and open-source ecosystems that are driving innovation and accelerating the adoption of Edge AI technologies.

Keywords: Edge AI, Embedded Systems, TinyML, AI Accelerators, Low-Power Inference, Model Optimization, Real-Time Processing.

1. Introduction

Edge Artificial Intelligence (Edge AI) refers to the deployment of machine learning models and inference capabilities on embedded processors or on-premises computing devices that operate independently of centralized cloud servers. Unlike cloud-based AI, where data must be continuously transmitted to remote infrastructures for processing, Edge AI performs computation locally at or near the data source. This paradigm significantly reduces network dependency, minimizes latency, enhances data privacy, and improves real-time responsiveness.

The need for on-device intelligence arises from the growing demand for low-latency, high-reliability decision-making in applications where even milliseconds matter. For example, an autonomous drone avoiding an obstacle or a self-driving car reacting to sudden movement cannot afford communication delays caused by cloud dependency. Similarly, in healthcare and surveillance systems, processing sensitive information locally mitigates data leakage risks while maintaining continuous functionality in bandwidth-limited or offline environments.

Energy efficiency also plays a critical role in Edge AI system design. Since many edge devices are battery-powered, they require optimized architectures that balance performance and power consumption. Techniques such as just-in-time processing, power-aware scheduling, and hardware-software co-design ensure efficient operation under constrained energy budgets while maintaining safety-critical performance.

Collectively, these advancements mark a transformative shift in intelligent computing. Autonomous robots and drones now rely on onboard AI for navigation, healthcare wearables enable real-time monitoring and anomaly detection, and industrial IoT systems leverage predictive analytics for proactive maintenance. As a result, embedded systems have emerged as the foundation of the Edge AI ecosystem—bringing intelligence closer to the source and enabling faster, more secure, and more efficient computation across diverse applications.

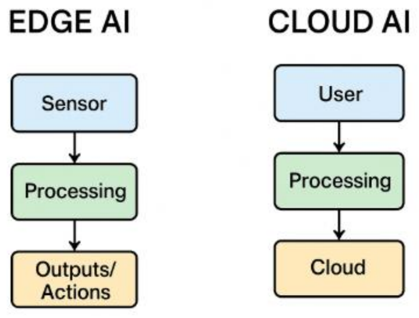

Figure 1. Block Diagram of Edge AI vs Cloud AI

2. Hardware Architecture for Edge AI

Embedded Edge AI systems rely heavily on optimized hardware platforms to deliver efficient, low-latency, and energy-aware computation [1]. The choice of hardware directly influences an edge device's performance, power consumption, scalability, and suitability for specific AI workloads. The most common hardware classes supporting Edge AI include Microcontroller Units (MCUs), System-on-Chips (SoCs), AI Accelerators, and Neuromorphic Processors - each offering unique trade-offs in performance and efficiency.

Table 1. Comparison of Hardware Architecture for Edge AI

Category	**Examples**	**Strengths**	**Limitations**	**Typical Applications**
Microcontrollers (MCUs)	STM32, ESP32	Ultra-low power, cost-effective, small form factor	Limited memory and computing power, suitable only for small AI models	Keyword spotting, anomaly detection in IoT sensors
System-on-Chips (SoCs)	NVIDIA Jetson, Qualcomm Snapdragon, Google Coral	High performance with integrated CPU/GPU/AI engines, versatile software support	Higher power consumption, thermal management needed	Robotics, computer vision, and autonomous navigation
AI Accelerators	Intel Movidius VPU, Google Edge TPU, MediaTek NPU, Huawei NPU, FPGAs	Optimized for AI inference, energy-efficient, and FPGA allows reconfigurability	Limited general-purpose functionality requires a supporting host processor	Smart cameras, speech recognition, industrial IoT
Neuromorphic Processors	Intel Loihi, BrainChip Akida	Brain-inspired design, ultra-low power, event-driven processing	Still experimental, limited ecosystem, and tool support	Next-gen wearables, adaptive robotics, edge cognition

a. Microcontroller Units (MCUs)

Microcontrollers such as STM32 and ESP32 are designed for lightweight, low-power AI applications like keyword spotting, anomaly detection, and sensor data analysis. These devices enable TinyML deployment, running compact neural networks within constrained memory and energy budgets. Their small footprint and cost-effectiveness make them ideal for large-scale IoT deployments. However, MCUs have limited computational capacity and memory, restricting them to simple inference tasks rather than complex deep learning workloads.

b. System-on-Chips (SoCs)

SoCs integrate CPUs, GPUs, NPUs, and DSPs on a single chip, enabling efficient on-device execution of computer-intensive AI models. Platforms such as NVIDIA Jetson, Qualcomm Snapdragon, and Google Coral offer high-performance inference for real-time applications in robotics, autonomous navigation, and intelligent vision [2]. While SoCs provide a balance between flexibility and computational strength, their higher power consumption and heat dissipation requirements make them less suitable for ultra-constrained environments.

c. AI Accelerators

Dedicated AI accelerators - such as Intel Movidius VPUs, Google Edge TPUs, MediaTek NPUs, Huawei Ascend, and FPGAs - are specifically designed to optimize AI inference performance at minimal energy cost. They deliver high throughput and efficiency for deep neural networks through specialized dataflow architectures. FPGAs offer reconfigurability, allowing developers to tailor hardware behavior to specific workloads. Despite their advantages, AI accelerators often depend on host processors for general-purpose tasks and may require specialized toolchains for deployment.

d. Neuromorphic Processors

Emerging neuromorphic processors, such as Intel Loihi and BrainChip Akida, emulate the behavior of biological neural systems using spiking neural networks (SNNs). These event-driven architectures achieve ultra-low power consumption and support real-time adaptive learning. Although still in the experimental phase, neuromorphic hardware holds immense promise for next-generation edge cognition, enabling applications in robotics, autonomous systems, and sensory processing where continuous adaptation and energy efficiency are critical.

e. Trade-offs and Limitations

Each hardware class presents distinct trade-offs between computational power, latency, and energy efficiency. While MCUs excel in power conservation, SoCs and AI accelerators provide superior performance at the cost of increased energy and thermal overhead. Neuromorphic processors, although promising, are constrained by limited tool support and ecosystem maturity. Additionally, challenges such as heat dissipation, battery endurance, and form-factor constraints remain central to designing sustainable and deployable embedded AI hardware [3].

In summary, the hardware foundation of Edge AI is defined by a continuous balance between performance and efficiency. The diversity of available platforms enables a wide range of applications - from low-power IoT sensors to high-performance autonomous systems - laying the groundwork for intelligent, adaptive, and energy-efficient computing at the edge.

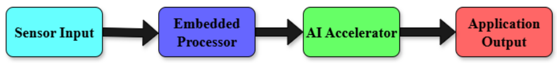

Figure 2: Diagram of an Edge AI system Architecture

3. Software Stack and Embedded AI Framework

The software ecosystem for Embedded Edge AI serves as the foundation for enabling efficient, real-time intelligence on resource-constrained devices. Unlike traditional AI implementations that rely on centralized cloud infrastructures, embedded AI frameworks integrate directly into lightweight operating systems and real-time environments, ensuring deterministic execution and optimized resource utilization.

At the base level, these systems are built upon real-time operating systems (RTOS) such as FreeRTOS and Zephyr, as well as Linux-based edge platforms, which support multitasking and device-level scheduling for time-critical operations. RTOS platforms are particularly suited for deterministic workloads like sensor fusion and control loops, while Linux-based frameworks provide greater flexibility for handling complex tasks such as image processing and neural network inference.

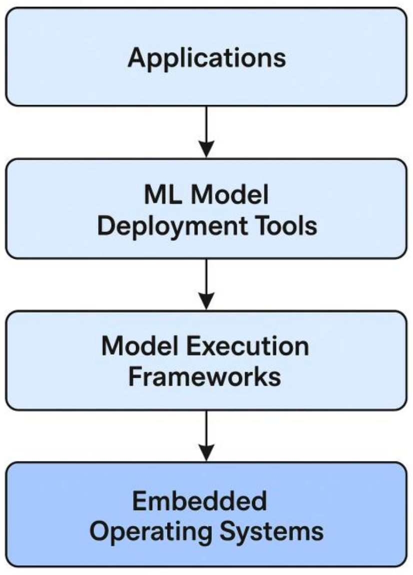
Applications
ML Model Deployment Tools
Model Execution Frameworks
Embedded Operating Systems

Figure 3: Software Stack Layers

Above the OS layer, model implementation frameworks form the core of Edge AI software. Frameworks such as TensorFlow Lite (TFLite) and TFLite Micro have become the de facto standards for deploying machine learning models on microcontrollers (MCUs) and SoCs. TFLite provides model quantization, conversion, and optimization for embedded environments, while TFLite Micro extends these capabilities to ultra-low-power devices with minimal RAM and flash memory. Similarly, ONNX Runtime enables cross-platform interoperability, allowing developers to run pre-trained models across CPUs, GPUs, FPGAs, and NPUs with minimal modification.

End-to-end development platforms like Edge Impulse streamline the workflow of data collection, model training, and deployment. These tools provide visual interfaces and automation pipelines, allowing even non-expert users to design and deploy models efficiently. At the compiler level, vendor-specific optimization tools - such as Arm Ethos-N, Apache TVM, and Glow - translate and fine-tune AI models for hardware-specific acceleration using NPUs or custom AI cores. These compilers automatically apply transformations such as layer fusion, quantization, and pruning to maximize performance while minimizing power consumption.

In parallel, the broader AI ecosystem is adopting technologies such as containerization and virtualization to improve modularity and portability. These techniques, common in cloud computing, are increasingly being adapted for edge deployment to ensure scalability and simplified maintenance. The use of cross-compilation and real-time scheduling further enhances development flexibility, allowing AI workloads to be tested and deployed across diverse hardware architectures with minimal manual configuration.

Together, these components - operating systems, runtime frameworks, compilers, and toolchains - form a cohesive embedded software stack that supports portability, scalability, and performance across the Edge AI landscape. This integrated software ecosystem enables developers to deploy optimized, low-latency AI models on embedded platforms, thereby accelerating the realization of intelligent, energy-efficient, and autonomous systems at the edge.

Table 2. Comparison of Software Frameworks for Embedded Edge AI

Framework	Target Devices	Key Features	Advantages	Limitations
TensorFlow Lite (TFLite)	MCUs, SoCs, NPUs	Lightweight ML inference, model conversion	Widely supported, Google ecosystem	Limited advanced optimizations
TFLite Micro	Ultra-low power MCUs	TinyML, <100 KB RAM support	Runs on bare-metal or RTOS, open-source	Restricted to small models
ONNX Runtime	Cross-platform (CPU, GPU, FPGA, NPU)	Open Neural Network Exchange support	Interoperable, vendor-agnostic	Heavier footprint than TFLite Micro
Edge Impulse	MCUs, SoCs, IoT devices	End-to-end pipeline (data collection → deployment)	Low-code environment, easy for beginners	Less control for advanced customization
Apache TVM	SoCs, GPUs, FPGAs	Compiler stack for deep learning optimization	High-performance, hardware-aware tuning	Steeper learning curve
DeepC	MCUs, IoT platforms	C-based AI model compilation	Fast execution, lightweight	Smaller developer community
Arm Ethos-N	Arm-based NPUs	Hardware-accelerated AI execution	Optimized for Arm platforms	Proprietary, limited flexibility

4. Design Methodologies for Embedded Edge AI

Developing efficient Embedded Edge AI systems requires specialized design methodologies that balance performance, power efficiency, and real-time responsiveness. These systems must operate within stringent resource constraints while maintaining the accuracy and reliability required for critical edge applications. Achieving these objectives involves optimizing the AI model, hardware architecture, and software pipeline through coordinated and iterative design techniques.

A key design challenge lies in model compression, which enables large deep learning models to run effectively on resource-limited hardware. Techniques such as quantization, pruning, and knowledge distillation are widely used to reduce computational complexity and memory footprint [6]. Quantization decreases the numerical precision of weights and activations, leading to faster inference and lower energy consumption. Pruning eliminates redundant neurons or filters without significantly impacting model performance. Knowledge distillation transfers learned representations from a large, high-capacity model to a smaller one, preserving accuracy while improving deployment. In parallel, Neural Architecture Search (NAS) is increasingly used to automatically identify optimal model architectures tailored for specific embedded platforms, achieving the best trade-off between latency and energy efficiency.

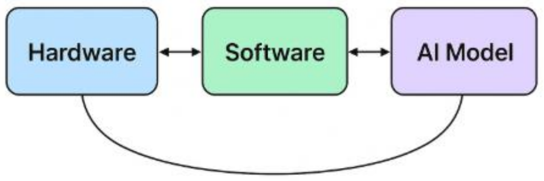

Figure 4: Design Methodologies for Embedded Edge AI

Another essential consideration is computational and memory optimization. Techniques such as tile-based computation divide workloads into smaller, manageable partitions that maximize the utilization of limited on-chip memory. Likewise, layer fusion merges multiple consecutive operations into a single computational kernel to minimize data transfer overhead. These methods enhance performance, particularly in low-latency and high-efficiency applications. Edge deployments using FPGAs and ASICs further benefit from these strategies by combining hardware-level acceleration with deterministic response times [7].

Compiler optimization also plays a critical role in bridging high-level models with low-level hardware. Frameworks such as MLIR, Glow, and XLA compile and transform neural network graphs into optimized, hardware-specific instructions. On microcontrollers, CMSIS-NN provides a library of highly optimized kernels that accelerate AI workloads while maintaining minimal memory usage [8]. These compiler-driven pipelines allow AI applications to be efficiently executed across heterogeneous devices, from

MCUs to SoCs and NPUs, while ensuring optimal throughput.

An emerging paradigm in this field is hardware–software co-design, which treats hardware, software, and AI model development as an integrated design cycle. Through co-design, neural network architecture, compilers, and hardware accelerators are optimized jointly to achieve significant gains in energy efficiency, latency reduction, and inference precision [9]. This methodology is particularly vital for miniaturized edge devices - such as wearables, sensors, and portable robotics - that demand extended battery life and computational agility.

Collectively, these design methodologies enable the creation of scalable, high-performance embedded AI systems capable of balancing model complexity, processing speed, and power efficiency. By integrating compression, compiler optimization, and co-design principles, developers can deliver AI solutions that are not only compact and energy-efficient but also robust enough to meet the demands of real-time edge intelligence across diverse application domains.

5. Applications and Case Studies

Embedded Edge AI has evolved beyond the research and prototyping stage and is now actively deployed across a wide range of real-world applications. By processing data locally rather than relying on cloud infrastructures, Edge AI enables low-latency inference, enhanced privacy, and greater operational autonomy. This distributed intelligence model is revolutionizing domains such as smart vision, industrial IoT, healthcare, and autonomous systems.

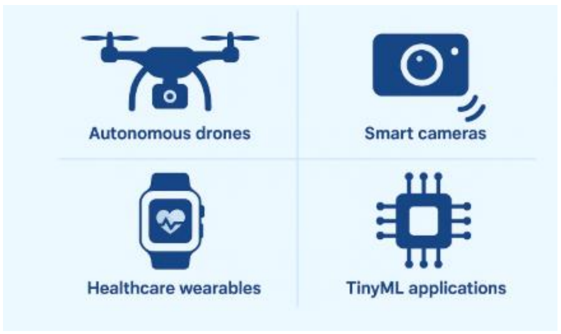

Figure 5: Application domains of Edge AI

In the field of smart vision, devices like OpenMV and ESP-EYE integrate on-device person detection and object recognition capabilities, allowing them to perform local video analytics without cloud connectivity. These embedded vision systems are widely used in smart home automation, security surveillance, and industrial inspection, offering real-time responses while ensuring data privacy. By eliminating the need to transmit images or video streams to remote servers, these systems achieve faster decision-making and maintain user confidentiality.

A representative example of highly efficient Edge AI deployment is TinyML, a subset of machine learning focused on ultra-low-power embedded systems [10]. TinyML frameworks allow neural networks to operate on microcontrollers with extremely limited computational resources. Applications range from voice recognition and keyword spotting (as used in Google’s on-device speech interfaces) to predictive maintenance in industrial IoT, where vibration sensors detect early signs of mechanical failure. Such implementations demonstrate that meaningful AI inference can be achieved using minimal power and memory, making it viable for large-scale embedded deployments.

Table 3. Applications and Case Studies of Embedded Edge AI

Application	Example Devices	Hardware/Software Stack	Datasets / Benchmarks
Smart Cameras (Person Detection)	OpenMV, ESP-EYE	MCU/SoC + TensorFlow Lite Micro, Edge Impulse	Visual Wake Words (VWW), COCO Tiny
TinyML – Keyword Spotting	Arduino Nano 33 BLE Sense, STM32	TFLite Micro, CMSIS-NN	Google Speech Commands Dataset
TinyML – Anomaly Detection in Vibration Sensors	ESP32, ARM Cortex-M MCUs	Edge Impulse, TVM	EdgeML vibration datasets, Industrial IoT datasets
Healthcare Wearables (Arrhythmia Detection, Fall Detection)	Smartwatches, ECG patches	SoC + TFLite, ONNX Runtime	MIT-BIH Arrhythmia Dataset, MobiAct Fall Detection Dataset
Autonomous Drones (Visual Navigation)	NVIDIA Jetson Nano, Qualcomm Snapdragon Flight	CNNs on SoCs/NPUs with TFLite, TVM	Drone Navigation datasets, MLPerf Tiny Vision Tasks

In healthcare, Edge AI has become a key enabler of intelligent monitoring systems. Wearable devices equipped with embedded AI models can continuously analyze physiological signals to detect anomalies such as arrhythmias or falls. These systems process data locally - using accelerometer or ECG signals - ensuring real-time detection and maintaining data security by minimizing cloud dependency. Such devices are critical in enhancing patient safety, particularly for elderly or high-risk individuals who require continuous monitoring without compromising privacy.

Another significant area of application is autonomous robotics, particularly drones and mobile robots that leverage onboard convolutional neural networks (CNNs) for real-time visual navigation and obstacle avoidance [11]. By performing inference locally, these systems can make split-second navigation decisions without the latency of cloud communication, enhancing both safety and autonomy. This capability is essential in mission-critical environments such as disaster response, surveillance, and precision agriculture.

To evaluate and benchmark these edge-based systems, standardized frameworks such as MLPerf Tiny, Visual Wake Words (VWW), Speech Commands, and EdgeML are used for performance validation and model optimization [12]. These benchmarks facilitate comparability across hardware platforms, guide optimization strategies, and encourage innovation in embedded AI design. Collectively, these case studies illustrate how Embedded Edge AI is transforming modern computing by enabling intelligence to reside directly within devices. Its ability to deliver real-time, privacypreserving, and energy-efficient decision-making continues to expand its role in smart environments, autonomous systems, and next-generation connected ecosystems.

6. Challenges and Future Research Directions

Despite the rapid advancements in Embedded Edge AI, several technical and operational challenges continue to hinder their widespread adoption. One of the most critical issues lies in ensuring real-time reliability for safety-critical systems such as autonomous vehicles, medical devices, and industrial automation [13]. These systems demand precise synchronization between AI inference and control mechanisms, where even minor latency or inference errors can have serious consequences. Designing AI architectures that guarantee deterministic behavior under dynamic workloads remains a formidable challenge.

Another key research direction involves continuous learning and model adaptation. Current embedded models are typically static once deployed, limiting their ability to adapt to new data or evolving environments. Approaches such as federated learning and lifelong learning have emerged to address this limitation by enabling decentralized model retraining without directly transferring raw data [14]. These techniques not only preserve privacy but also extend the operational lifespan of AI-enabled devices by allowing them to learn incrementally from distributed sources.

Security and privacy also remain major concerns in Edge AI deployments. Embedded systems must be safeguarded against adversarial attacks, model inversion, and data tampering, all of which can compromise decision integrity. Implementing secure boot mechanisms, trusted execution environments, and encrypted inference pipelines is essential to ensure end-to-end system integrity. The integration of hardware-based security modules further strengthens trust in edge deployments, particularly in sensitive domains such as defense and healthcare.

From a hardware perspective, there is a growing trend toward domain-specific architectures (DSAs) and reconfigurable hardware such as FPGAs and NPUs, designed to optimize specific AI workloads [15]. These platforms promise improved efficiency and flexibility but introduce new challenges related to standardization and interoperability. Currently, the lack of unified software toolchains and open benchmarks hampers rapid development and cross-platform compatibility. Future research must focus on open-source ecosystems, standardized evaluation frameworks, and unified compiler infrastructures to accelerate innovation and reduce fragmentation across the embedded AI landscape.

Overall, the future of Embedded Edge AI will depend on progress in robustness, adaptability, security, and standardization. Addressing these challenges is essential for realizing fully autonomous, intelligent, and trustworthy systems that can operate seamlessly at the network edge.

7. Conclusion

The evolution of Embedded Edge AI represents a convergence of hardware innovation, software optimization, and co-design methodologies. The synergy between these domains has enabled the deployment of intelligent systems capable of real-time processing, low power consumption, and enhanced autonomy. Hardware platforms - ranging from microcontrollers and SoCs to neuromorphic processors - form the backbone of this transformation, while lightweight frameworks such as TensorFlow Lite, ONNX Runtime, and Edge Impulse bridge the gap between model development and deployment.

Design methodologies such as quantization, pruning, and hardware, software co-design continues to refine system efficiency, allowing AI models to function effectively on resource-limited devices. These innovations are already visible across applications in smart vision, wearable technology, industrial IoT, and robotics, where embedded intelligence has become an operational necessity. Looking ahead, the next generation of Edge AI systems will prioritize on-device learning, interoperable architecture, and secure real-time computation. Research will increasingly focus on reliability, trustworthiness, and energy-efficient AI design, ensuring that embedded systems not only compute intelligently but also operate responsibly within critical environments. As these technologies mature, Embedded Edge AI will become a cornerstone of intelligent computing - merging speed, privacy, and adaptability into the very fabric of future connected ecosystems.

References

[1] Mazzia, V., Khaliq, A., Salvetti, F. and Chiaberge, M., 2020. Real-time apple detection system using embedded systems with hardware accelerators: An edge AI application. Ieee Access, 8, pp.9102-9114.

[2] V. Shankar, "Machine Learning for Linux Kernel Optimization: Current Trends and Future Directions," International Journal of Computer Sciences and Engineering, vol. 13, no. 3, pp. 56–64, 2025.

[3] Sipola, T., Alatalo, J., Kokkonen, T. and Rantonen, M., 2022, April. Artificial intelligence in the IoT era: A review of edge AI hardware and software. In 2022 31st Conference of Open Innovations Association (FRUCT) (pp. 320-331). IEEE.

[4] V. Shankar, M. M. Deshpande, N. Chaitra, and S. Aditi, “Automatic detection of acute lymphoblastic leukemia using image processing,” in *Proc. IEEE Int. Conf. Adv. Comput. Appl. (ICACA)*, Coimbatore, India, 2016, pp. 186–189, doi: 10.1109/ICACA.2016.7887948.

[5] Imran, H.A., Mujahid, U., Wazir, S., Latif, U. and Mehmood, K., 2020. Embedded development boards for edge-AI: A comprehensive report. arXiv preprint arXiv:2009.00803.

[6] Marwedel, P., 2021. Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things (p. 433). Springer Nature.

[7] Mendez, J., Bierzynski, K., Cuéllar, M.P. and Morales, D.P., 2022. Edge intelligence: concepts, architectures, applications, and future directions. ACM Transactions on Embedded Computing Systems (TECS), 21(5), pp.1-41.

[8] Soro, S., 2021. TinyML for ubiquitous edge AI. arXiv preprint arXiv:2102.01255.

[9] V. Shankar, “Edge AI: A comprehensive survey of technologies, applications, and challenges,” in *Proc. 1st IEEE Int. Conf. Adv. Comput. Emerg. Technol. (ACET)*, Ghaziabad, India, 2024, pp. 1–6, doi: 10.1109/ACET61898.2024.10730112.

[10] Adami, D., Ojo, M.O. and Giordano, S., 2021. Design, development and evaluation of an intelligent animal repelling system for crop protection based on embedded edge-AI. IEEE Access, 9, pp.132125-132139.

[11] Hao, C., Dotzel, J., Xiong, J., Benini, L., Zhang, Z. and Chen, D., 2021. Enabling design methodologies and future trends for edge ai: Specialization and codesign. IEEE Design & Test, 38(4), pp.7-26.

[12] V. Shankar, R. Gautham, and Vedaprakashvarma, “Automated

traffic signal for hassle-free movement of ambulance," in *Proc. IEEE Int. Conf. Electr., Comput. Commun. Technol. (ICECCT)*, Coimbatore, India, 2015, pp. 1–5, doi: 10.1109/ICECCT.2015.7226114.

[13] Zhang, X., Lu, H., Hao, C., Li, J., Cheng, B., Li, Y., Rupnow, K., Xiong, J., Huang, T., Shi, H. and Hwu, W.M., 2020. SkyNet: a hardware-efficient method for object detection and tracking on embedded systems. Proceedings of Machine Learning and Systems, 2, pp.216-229.

[14] Letaief, K.B., Shi, Y., Lu, J. and Lu, J., 2021. Edge artificial intelligence for 6G: Vision, enabling technologies, and applications. IEEE journal on selected areas in communications, 40(1), pp.5-36.

[15] Yadav, A.B., 2023. Gen AI-driven electronics: innovations, challenges and future prospects. In International congress on models and methods in modern investigations (pp. 113-121).

[16] N. Muralidhar, V. Shankar, and K. Narendra, "Assessing the effectiveness of contrast limited adaptive histogram equalization with DeepLabV3+ in brain tumor segmentation," in Proc. IEEE Int. Conf. Knowl. Eng. Commun. Syst. (ICKECS), Chickballapur, India, 2025, pp. 1–7, doi: 10.1109/ICKECS65700.2025.11035745.

[17] Belabed, T., Coutinho, M.G.F., Fernandes, M.A., Sakuyama, C.V. and Souani, C., 2021. User driven FPGA-based design automated framework of deep neural networks for low-power low-cost edge computing. IEEE Access, 9, pp.89162-89180.

[18] V. Shankar, "Chip architecture for AI – A comprehensive survey of design principles, technologies, and implementations," in *Proc. 2nd IEEE Int. Conf. Signal Process., Commun., Power Embedded Syst. (SCOPES)*, Paralakhemundi Campus, Centurion Univ. Technol. Manage., Odisha, India, 2024, pp. 1–6, doi: 10.1109/SCOPES64467.2024.10991015.

[19] Paissan, F., Ancilotto, A. and Farella, E., 2022. Phinets: A scalable backbone for low-power ai at the edge. ACM Transactions on Embedded Computing Systems, 21(5), pp.1-18.

[20] Chen, Y., Zheng, B., Zhang, Z., Wang, Q., Shen, C. and Zhang, Q., 2020. Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions. ACM Computing Surveys (CSUR), 53(4), pp.1-37.

[21] V. Shankar, "Advancements in AI-Based Compiler Optimization Techniques for Machine Learning Workloads," International Journal of Computer Sciences and Engineering, vol. 13, no. 3, pp. 70–77, 2025.

[22] Kallimani, R., Pai, K., Raghuwanshi, P., Iyer, S. and López, O.L., 2024. TinyML: Tools, applications, challenges, and future research directions. Multimedia Tools and Applications, 83(10), pp.29015-29045.