Journal of Adaptive Learning Technologies
ISSN **2997-3902** (Online)
Vol. 3, No. 1, Jan 2026,
Available online at: https://scientificbulletin.com/index.php/JALT

EMBEDDED SYSTEMS AI ON MICROCONTROLLERS LIGHTWEIGHT AI MODELS FOR REAL-TIME INFERENCE ON RESOURCE-CONSTRAINED MICROCONTROLLERS IN EMBEDDED SYSTEMS

Sarmad Hamad Ibrahim Alfarag
Electrical Engineering Department, Wasit University, Republic of Iraq

Article information:
Manuscript received: 05 Nov 2025; **Accepted:** 06 Dec 2025; **Published:** 30 Jan 2026

Abstract: This paper addresses the efficacy and significance of using bare-bones AI frameworks in the detection of wear, tear and malfunctions that affects the embedded system. By employing pruning, quantum transformation, knowledge extraction mechanisms, we were able to adapt deep learning models to increase the efficiency of peripheral devices, providing high accuracy and low latency, while maintaining small memory and low power consumption. The use of bare-bones AI technologies is not only based on cloud architecture but also supports data privacy, system independence, and operational reliability in industrial systems, respiratory healthcare devices, and measurement devices, the study offers an integrated vision of a real-time fault detection system, and is characterized by being scalable and deployable through a large set of embedded applications. With the large wave in the development of embedded healthcare devices that are linked to artificial intelligence and raise the possibility of building more complex and broader intelligent edge systems, this research also paves the way for discovering fault detection solutions that can be deployed in the embedded system. Edge AI's ability to access data has become evident, reducing latency, enhancing privacy, and consuming less power within the system. This study focuses on the importance of edge AI in embedded systems, the mechanisms of software operation, and how to design a proposal that improves performance, particularly in practical applications such as healthcare devices (Healthcare devices for breathing and heart rate monitoring) the Internet of Things (IoT), and robotics. The XGBOOST, CNN, LSTM, and Random Forest algorithms were trained on the same data for training and testing to ensure neutrality in comparison and evaluation, and normalization processes were performed to measure differences in speed, accuracy, and memory, The experiments performed in the proposed model were conducted using Python 3.10 and the NumPy, Pandas, and Scikit learn libraries. The training was conducted on Linux computers with the use of Intel i7 processor and 16 GB of RAM Deployment was carried out using Arduino Portenta and Raspberry Pi Pico. The 1D-CNN model includes a Conv1D layer with 64 filters and 3 cores, followed by Max Pooling, Flatten, and Dense for classification. The LSTM model uses the Adam model layer as an optimizer and Cross Entropy as a loss, with 50 Epoch and a batch size of 32. In the study model, the models were evaluated using: F1-score m Recall, Precision, Accuracy, and an ambiguity matrix to diagnose classification accuracy and distinguish between normal and abnormal cases. The goal is to enable intelligent decision-making with less time and energy, ensuring the reliability and safety of industrial software and applications. This study also recommends easing the challenges facing modern trends, such as scheduling. The tasks, reliable and

secure reasoning, while working on the need to deepen operational harmony with the open-source system, which relies on peripheral artificial intelligence.

Keywords: Edge-enabled, Artificial Intelligence, TinyM, Machine learning, Healthcare devices.

1. Introduction

Artificial intelligence (AI) has completely changed the safety monitoring and predictive management in the presumed embedded systems. It has indicated a system can detect and predict faults, failures and disruptions in an industrial system. The industrial operating systems need maximum flexibility and consistency in error or fault detection for a perfect operation, minimized damage, and reduced downtime. This is because of the absence of real-time responsiveness of local cloud technologies, which may create security challenges and eventually depend on incompatible tool in the automated applications. This study sets to focus on installing lightweight AI models in the fixed systems to allow real-time edge-level fault discovery. It depends on model development tools, which include knowledge distillation, quantization, pruning and fixing AI algorithms into a fixed hardware systems without a reduction in performance. Edge AI models demonstrate a great ability to achieve high accuracy in detecting faults and problems, while reducing the time required to infer the presence of a fault, which leads to low power consumption. They also enable and ensure the reliability and security of application operation, because traditional fault and problem detection methods are characterized by being unsuitable for immediate scenarios due to delays in detecting problems, resulting in security risks, and their dependence on communication. Edge AI models show an ideal solution in their ability to make decisions quickly, securely, and independently of the source. Despite the advantages that edge AI models have, they face challenges in embedded system hardware in working on the consistency and balance of the model in rapid inference and using less power. Previous work that achieved high accuracy and reliability did not pay attention to the limitations, and other studies suggested deploying AI models that do not feature development that is compatible with real-time. Therefore, this study will focus on producing a model that is dependent on previous studies and will attempt to combine a model design with efficient optimization approaches. Similarly, real time performance trial was carried out on the devices through the use of relevant field data in order to address the gap with the providsion of a framework that will deal with the deficiencies in error detection within the fixed systems.

1.1 Objectives of Study

The objectives of this study are:

- To enhance and deploy artificial intelligence models in fixed systems.
- To notice errors and offer precise inferences.
- To make intellectual choices in record time with the use of edge tools with low computational resources and strength.
- To forecast faults and potential distractions that impact embedded systems.
- To reduce the effect of problems on the system as a whole.
- To decrease the downtime and self-management and to ensure maximum consistency.

1.2 Importance of the Study

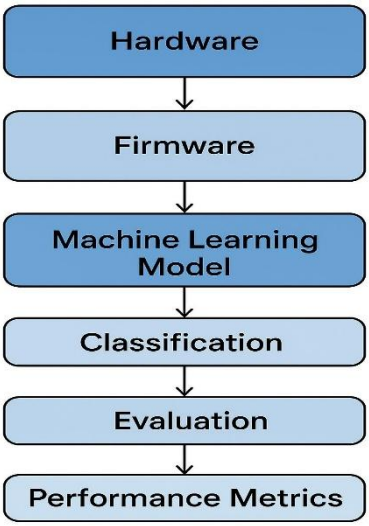
The following can be considered as the significance of the study:

- To offer artificial intelligence copies that are characterized by consistency and

precision in the detection of errors, problems and faults that often cause failures in the operating system.

- Adopting a model optimization like knowledge transfer, pruning and quantization to decrease model size and processing load.
- The testing and application of models on the newest fixed platforms, especially in the domain of smart health systems, owing to their capability to detect heart sleep disorders, and to allow for data analysis in case of a detection of a problem.

Figure (1) Diagram of AI system Architecture



1.3 TinyML Framework

Compressing machine learning models is one of the important aspects of building TinyML systems, that will enable the implementation of difficult models on resource-constraint platforms like the IoT and microcontrollers devices. Compression mechanisms are dependent on the perception that the internal structure of the models whether it is based on deep neural networks like 1D-CNN and LSTM and learning models like the Random Forest and XGBoost. Python 3.10 as well as the Numpy, pandas, Scikit learn libraries were used in the process of the expirements and the training conducted with Linux Computers, Intel i7 processor and 16 GB RAM capacity. The deployment conducted through Arduino Portenta and Raspberry Pi Pico. The 1D-CNN model comprises of a Conv1D layer with 64 filters and 3 cores, followed by MaxPooling, Flatten, and Dense for arrangement. The LSTM type applies the Adam model layer as an optimizer with 50 Epoch and a batch size containing 32.

2. Literature Review

Artificial intelligence (AI) has been expanding in scope of error discovery systems. This expansion is derived by the rigid nature of traditional statistics signal processing approaches, which seem restricted in their capability to deal with tricky, dynamic and non-linear ad flexible system patterns [1] in the same vein, AI techniques specifically machine learning and the deep learning and has allow the development of more dynamic and precise anomaly and error diagnosis structure On the other hand, [2] Cloud-based AI has been broadly been used for incongruity detection and fault identifying owing to its consistency in computational capacities [3]. Network dormancy problems and data confidentiality challenges and the need for regular connectivity may hinder their efficacy in the implementation of real-time scenarios [4]. Some of the merits of this approach, is of the reduced bandwidth and data confidentiality protection and it was highlighted by the growth

of the lightweight convolutional neural networks (CNNs) and the reoccurring neural networks (RNNs) [5][6]. These sorts of networks can offer an enhanced processing, greater precision, and reduced power usage. In addition, AI procedures can play an essential role in the improvement methods like the quantization, pruning and the focus on model-specific scaling and computational load decrease. A similar study by [7] which focused on the ability to quantize convolutional neural networks to attain genuine inference on an ARM platform with little power loss [8].

Compressing machine learning models is one of the crucial parts of building TinyML structures, as it will enable the implementation of difficult models on resource-constrained platforms which include microcontrollers and IoT tools. These algorithms may differ in their form of data presentation and approach to acquisition methods. LSTM may depend on processing serial sequences and its capability to maintain hold data overtime, while ID-CNN may be appropriate in the extraction of localized features form the serial signals like the vibrations sound. Random Forest, as indicated may rely on the integrated impact of a huge number of trees to form an overall choice, while XGBoost can be differentiated by its capability to build a successful model whose deficiencies can be addressed incrementally, as a result of its efficacy and greater accuracy.

Figure (2)

CNN Algorithm

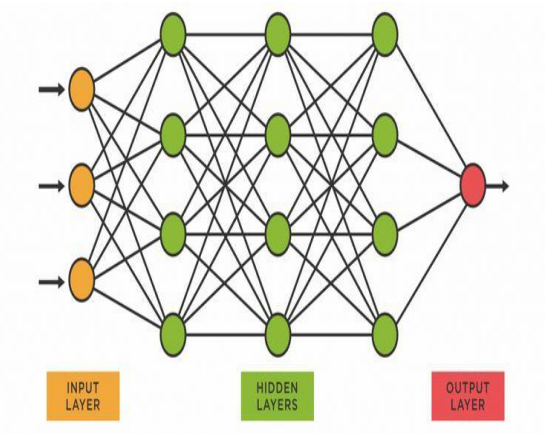
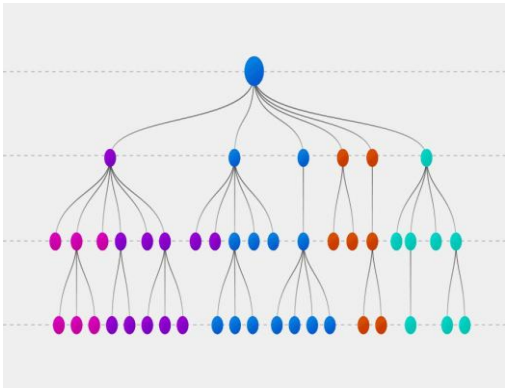


Figure (3)

Random Forest Algorithm



One of the most projecting approaches in the reduction of the size of these models is called pruning, which include the elimination of weights or connections with the least effect after the exercise. The approach can be systematic like deleting the complete filter units in 1D-CNN networks or reduction in the number of components in LSTM levels. In the context of the tree models such as the Random Forest and XGBoost, tree pruning can be used by removing nodes with little effect or grouping unrelated branches to minimize the computational difficulty without tempering the accuracy. Quantum changes are another important step in narrowing and the acceleration of the model implementation. The process may convert weights from floating-point format to integers with a precision of 16, 8, or even 4 bits, and it will crucially reduce space, power and power consumption. Network like the LSTM and ID-CNN are more prone to quantization as a result of the sensitive nature of their weights. Therefore, a sensible quantization transformation is often used during training to maintain the level of fine-tuning. In contrast, tree models such as XGBoost and Random Forest are easier to perform direct quantization due to their simpler separation barriers. Additional approaches such as Huffman coding can also be incorporated to compress the most frequently occurring weight values, which helps to reduce the final model size without any information degradation.[10]

Figure (4) XGBoost Algorithm

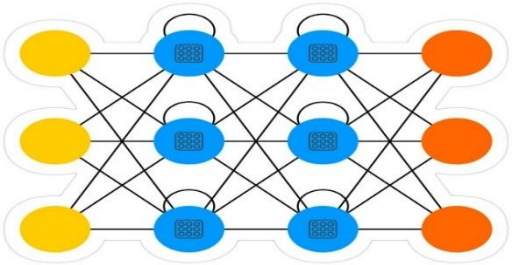
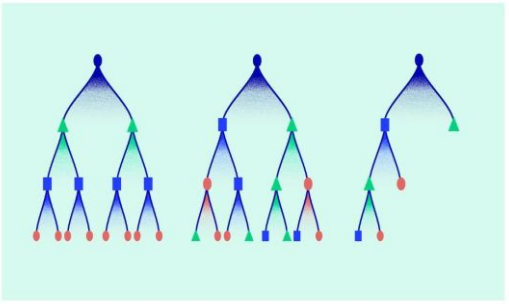


Figure (5) LSTM Algorithm



2.1 Practical Aspect

In this study, an artificial intelligence model was designed with the ability to detect faults and problems in the operating system early, predict failures and malfunctions before they occur to reduce downtime and self-maintenance, ensuring higher reliability. This aims to improve system performance by developing models with high accuracy and reliability, and lower power and memory consumption. In this section, we will collect the data, then process and normalize it for deployment on embedded platforms and extraction of results. The first step is data collection and organization. [11][10].

Table (1) shows the data types:

Device sensitivity data
Temperature
Press
Vibration
Voltage or current
Number of operating cycles or hours
Previous errors
Data labeling

2.1.1 Data Collection

Data was collected at a sampling rate of 50-100 Hz per second per reading. Two conditions, normal and abnormal, were observed, with 1000 samples per category. The data were reviewed to remove anomalous signals and ensure the homogeneity of the measurement environment, such as lighting, movement, and sensor stability.

2.1.2 Study Model Environment

The experiments performed in the proposed model were conducted using Python 3.10 and the NumPy, Pandas, and Scikit learn libraries. Training was performed on Linux computers with an Intel i7 processor and 16 GB of RAM. Deployment was carried out using Arduino Portenta and Raspberry Pi Pico.

Table (2) Machine learning algorithms for processing raw data in real time.

Raspberry Pi Pico	Microcontroller	Algorithms Facilities
Pulse Sensor PPG	Pulse Sensor	Heart Rate Measurement
Respiratory Belt	Respiration Sensor	Respiratory Rate Measurement
LED/ Buzzer	Actuator	Alerts & Notification

2.1.3 Model Structure

The 1D-CNN model includes a Conv1D layer with 64 filters and 3 cores, followed by

MaxPooling, Flatten, and Dense for classification. The LSTM model uses the Adam model layer as an optimizer and Cross Entropy as a loss, with 50 Epoch and a batch size of 32.

2.1.4 E valuation Criteria

In the study model, the models were evaluated using: F1-score m Recall, Precision, Accuracy, and an ambiguity matrix to diagnose classification accuracy and distinguish between normal and abnormal cases.

2.1.5 Real-World Testing Phase

The PPG sensors and breathing belt were connected to the microcontroller and directly tested on the model. Three hundred real-world samples were observed and compared with the model's results. The model demonstrated a rapid response with an average inference time of less than 20 ms on the Raspberry Pi Pico.

Real data was used, obtained from actual devices, industrial and medical devices connected to sensors, via the NASA Turbofan Engine Dataset and data from the UCI Machine Learning Repository. Simulated data was generated based on physical laws and device specifications. The Python library, along with libraries such as NumPy and SimPy, were used to generate the data. [13]

2.2 Data Preparation and Sorting

The CSV Numpy number of data was deleted using Python. With regards to data preparation, the first step is data cleaning to delete missing and anomalous values. This is followed by data standardization. The data was separated into 70% training, 15% confirmation, and 15% testing. The newest artificial intelligence technologies serve as a main technological dive across all similar levels and fields, principally in the field of smart health systems, owing to their capability to notice heart and sleep disarrays. The data is then read, processed, and alerted if any problem or disorder is detected. [14] The proposed model for detecting respiratory disorders and heart problems by monitoring pulse and respiration in real time is shown in Table (2):

Table (3) Represents the detection of sleep and respiratory disorders:

Heartbeat: 60 beats per second × 30 seconds sampling time = 1800 beats.
Respiration: 60 breaths per second × 30 seconds sampling time = 1800 breaths.
Classification
Normal
Abnormal
Irregular (Slow/Fast)
Irregular Respiration
Number of Samples: 1000 per category
Model Preparation: Normalize data between 1 and -1 for training and testing.

2.3 The proposed model

was implemented using Edge Impulse and TensorFlow. The version was trained on the same dataset using feature matching and the same type of convolutional neural network (CNN). The TensorFlow model took longer due to initial unsuccessful trials.

By combining pruning, quantum transformation, and Huffman coding, the size of LSTM models, 1D-CNNs, and tree models can be reduced up to a factor of thirty compared to their original size, while maintaining high levels of refinement, making them suitable for use on very limited hardware. Special frameworks have been developed to support TinyML, such as TensorFlow Lite Micro, designed for Cortex-M processors. It allows for

the execution of 1D-CNN and LSTM networks after condensation without the need for variable memory, ensuring stable performance and minimal power consumption. The Edge Impulse platform provides a comprehensive environment for data collection and the emulation of deep, tree-like models, which are then converted into condensed C++ code optimized for the EON Neural Compiler. uTensor facilitates the creation of small models that can be directly loaded onto microcontrollers. STMicroelectronics offers auxiliary tools like NanoEdge AI Studio and X-Cube-AI to support models such as 1D-CNN, Random Forest, and XGBoost on STM32 platforms and convert them into executable C code. A low-cost PPG heart rate sensor and a breathing rate monitor are also available. Raspberry Pi Pico Microcontroller with TinyML Support Board

2.4 TinyML Working Mode

TensorFlow Lite Micro: This is a version of TensorFlow Lite designed to optimize the TensorFlow model for high-resolution Cortex-M and maintain high-speed transfer. TensorFlow Lite Micro is simplified to minimize uncommon features and reduce memory fragmentation. It also eliminates dynamic memory for enhanced memory performance.

Edge Impulse: This is an integrated approach to TinyML. Data can be collected via IoT devices, and features are focused on training and deployment within the model. This optimizes embedded devices, and the Edge-Enhanced Neural Interpreter (EON) is used to translate the trained network into code, removing inactive C++ machine learning operator signals. Recent studies indicate the various applications of TinyML, which comprises of sensing individuals through 1D-CNN networks, detecting industrial defects and sensor-based gestures, XGBoost and Random Forest are due to their maximum ability to process wave-derived elements. These applications are dependent upon similar platforms like the Raspberry Pi Pico, Arduino Nano 33 BLE Sense, and STM32, which serve as a balance between efficiency, power consumption and ESP32. The research also indicates that numerous TinyML computers can function at frequencies below 100 MHz and have less than 1 MB of SRAM, with the use of the ARM Cortex-M architecture. More powerful systems like the Raspberry Pi 4 and NVIDIA Jetson Nano are excluded because they are not strictly throttled platforms. UTensor creates C++ files from TensorFlow to produce models smaller than 2 KB. It consists of two parts: uTensor Core (the core) and the uTensor Library. The core contains the data structure, runtime, and optimized interfaces. The library includes default error handlers, context, allocators, arrays built on the core, and machine learning operations. It utilizes TFLITE, a borrowed model compression and optimization mechanism. AI Algorithms in the Proposed Model **(Table 4):**

Table (4) Artificial Intelligence Algorithm in proposed model.

1D- CNN	Fast & Simple	Lower Accuracy in Sequential data
Random Forest	High Accuracy	Requires more power
XGBoost	High Stability	Slow Training & high memory consumption
LSTM	Excellent for Time Series Data	Requires a GPU for Large Scale Training

2.5

Results Analyses

The first experiment was conducted using 1D-CNN to automatically extract features.

To read patterns from sequential data (conv1D):

Reducing data dimensions while preserving important features (MaxPooling 1D):

Classification: Normal or abnormal (Dense Output)

Table show the Classification of AI Algorithm Performance in the Proposed Model:

Removing unnecessary parts of the model (Pruning):

Reducing the precision of weight and computational quantization:

Using a pre-trained model and adapting it to a new task (Knowledge Transfer):

This represents one of the main constraints in the design Compact models utilize quantization, pruning, and knowledge distillation techniques to significantly reduce complexity and memory consumption. Pruning can remove neurons without having effect on the performance, on the transfer of knowledge to a more capable model and maintain the precision and consistency as well as improving deployment.

The projected model will provide high-precision and consistent early error detection. It can also be improved with the use of pruning quantization and knowledge transmission to decrease the total model size and enhance assumption. It can be attached on platforms like Arduino Portenta, Raspberry Pi Pico 2, and STEM 32 Nucleo H7. The 1D-CNN model is highly precise but it needs longer training. It provides faster inference with less memory. The model choice is based on a stability between precision and inference speed.

Table (5) Organization of the performance of AI Algorithms in projected model.

Model	Training Accuracy	Testing time	Training time	Inference
XGBOOST	%85	%80	1	0.5
Random Forest	%92	%89	5	1
LSTM	%90	%87	10	2
1D-CNN	%98	%95	50	3

Table 5 indicates a comparison and assessment of the performance of the AI algorithms applied in the proposed study model. It showed that XGBoost was much faster in terms of training and testing, but it had the lowest precision, which achieve only 85% of successful testing. This shows that it is more essential when speed is considered that precision.

Similarly, Random Forest can perform well in terms of speed and was a bit slightly faster than XGBoost, which achieves a success rate of 92%. This will make it a more appropriate option when accuracy is considered. The LSTM type, was less efficient owing to the nature of specialized network, which require a longer training and testing time. Its accuracy can be rated as approximately 90%. 1D-CNN, on the other hand, was the most accurate, achieving 98% accuracy. Its training was completed, and the remaining progress was left to the individual. Overall, 1D-CNN demonstrated superiority in terms of accuracy, while Random Forest offered the best, XGBoost was best suited for operations requiring battery speed, and LSTM showed the series to be better performing in the proposed computer model.

Table (6) Model Improvement

Techniques	Purpose
Pruning	Remove unnecessary part of model
Quantization	Reduce the precision of weight & computational
Knowledge Transfer	Used a pre trained model & adapts it to a new task

Table 6 show the Implementation on hardware Embedded:

- This is achieved by connecting the microcontroller to the sensors.
- The model is converted to a TensorFlow Lite or TinyML C array.
- The model is sent to obtain real-time classification.
- LED & Buzzer are activated to detect unwanted conditions.

3. Conclusion and Future Work

Artificial intelligence (AI) has certainly changed the safety and monitoring as well as predictive management in fixed systems. It has shown that a system can be effective in detecting and predicting errors, failures and disruptions in industrial systems. Industrial operating systems need high flexibility and consistency in fault detection for perfect operation, minimized damage and reduced downtime. This is owing to the lack of real - time responsiveness of traditional cloud technologies that can create security failures and depend on incompatible techniques in most automated applications.

This research exhibits the efficacy and significance of deploying bare-bones AI models in the detection of wear tear and system malfunctions which affects the fixed systems. By using quantum transformation, knowledge extraction mechanisms, and pruning, it was able to change deep learning models to enhance the efficacy of minor devices, low latency and provide high accuracy and as well maintain small memory and low power consumption. The application of bare-bones AI technologies is not only restricted to cloud architecture, but also supports data confidentiality. System reliance, and operational accuracy in industrial systems, and other healthcare facilities, can be integrated in the real-time application and integration vision. This is applicable by being scalable and can be deployed via a large set of fixed applications. With the large wave in the development of embedded healthcare devices that are linked to artificial intelligence and raise the possibility of building more complex and broader intelligent edge systems, this research also paves the way for discovering fault detection solutions that can be deployed in the embedded system.

Future work: The model can be extended to include SpO2 and ECG sensors, and lighter architectures such as MOBILENET-TINY can be tested. Power consumption analysis can also be added, along with improved data display interfaces and testing on human subjects under different conditions. The results obtained during the model's classification and deployment phases may depend on the data volume, be affected by noise and vibration measurements during movement, and some controllers with limited memory may hinder the model's use. It should be noted that the scope of this study does not apply to all medical scenarios.

References

1. **Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., & Hafid, A. S. (2023).** A comprehensive survey on TinyML. *IEEE Access, 11*.
2. **Deebak, B. D., & Al-Turjman, F. (2023).** EEI-IoT: Edge-enabled intelligent IoT framework for early detection of COVID-19 threats. *Sensors, 23.*
3. **Elhanashi, A., Dini, P., Saponara, S., & Zheng, Q. (2024).** Advancements in TinyML: Applications, limitations, and impact on IoT devices. *Electronics, 13*(17).
4. **Grzesik, P., & Mrozek, D. (2024).** Combining machine learning and edge computing: Opportunities, challenges, platforms, frameworks, and use cases. *Electronics, 13*(3).
5. **Heydari, S., & Mahmoud, Q. H. (2025).** Tiny machine learning and on-device inference: A survey of applications, challenges, and future directions. *Sensors, 25.*

6. **Immonen, R., & Hämäläinen, T. (2022).** Tiny machine learning for resource-constrained microcontrollers. *Journal of Sensors, 2022*(1).
7. **Jouini, O., Sethom, K., A. N., Aljohani, N., Alanazi, M. H., & Alanazi, M. N. (2024).** A survey of machine learning in edge computing: Techniques, frameworks, applications, issues, and research directions. *Technologies, 12*(6).
8. **Kelhälä, E. (2024).** *Evaluating hardware platforms for computer systems education* (BSc thesis).
9. **Kelhälä, E., Islam, R., & Milara, I. S. (2024).** [Demo] Building an open-source, open-hardware platform for IoT education for higher education. In *Proceedings of the 14th International Conference on the Internet of Things (IoT2024).*
10. **Li, F., & Wang, C. (2023).** Artificial intelligence and edge computing for teaching quality evaluation based on 5G-enabled wireless communication technology. *Journal of Cloud Computing, 12.*
11. **Paul, P. (2022).** Edge computing & educational systems: Towards advanced and intelligent learning—A conceptual overview. *International Journal of Information Science and Computing, 9.*
12. **Ray, P. P. (2022).** A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences, 34,* 1595–1623.
13. **Ren, J., Zhu, Q., & Wang, C. (2022).** Edge computing for water quality monitoring systems. *Mobile Information Systems, 2022,* 5056606.
14. **Saha, S. S., Sandha, S. S., & Srivastava, M. (2022).** Machine learning for microcontroller-class hardware: A review. *IEEE Sensors Journal, 22*(22), 21362–21390.