Human Activity Recognition on Embedded Devices: An Edge AI Approach

Graziele de Cássia Rodrigues[1][a] and Ricardo Augusto Rabelo Oliveira[2][b]
[1]*Computing and Systems Department, Universidade Federal de Ouro Preto, Joao Monlevade, Brazil*
[2]*Computing Department, Universidade Federal de Ouro Preto, Ouro Preto, Brazil*

Keywords: Human Activity Recognition, Embedded Devices, TensorFlow Lite, Real-Time, Edge AI.

Abstract: Human Activity Recognition (HAR) is a technology aimed at identifying basic movements such as walking, running, and staying still, with applications in sports monitoring, healthcare, and supervision of the elderly and children. Traditionally, HAR data processing occurs in cloud servers, which presents drawbacks such as high energy consumption, high costs, and reliance on a stable Internet connection. This study explores the feasibility of implementing human activity recognition directly on embedded devices, focusing on three specific movements: walking, jumping, and staying still. The proposal uses machine learning models implemented with LiteRT (known as TensorFlow Lite), enabling efficient execution on hardware with limited resources. The developed proof of concept demonstrates the potential of embedded systems for real-time activity recognition. This approach highlights the efficiency of edge AI, enabling local inferences without the need for cloud processing.

1 INTRODUCTION

Human Activity Recognition (HAR) involves identifying basic movements such as walking, running, jumping, and standing still (V. Sharma and Cano, 2024). This technology has applications in various areas, including sports monitoring, hospital care, elderly and child supervision, among others (Shubham Gupta and Deb, 2022). However, processing these data, commonly performed on cloud servers, poses challenges such as high energy consumption and reliance on Internet connectivity (Bidyut Saha and Roy, 2024).

Traditionally, machine learning algorithms, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and long short-term memory networks (LSTMs), have been used for human activity classification (Waghchaware and Joshi, 2024). However, these methods require significant computational resources, making them challenging to implement on resource-constrained devices. A promising approach to overcoming these challenges is edge computing, where data processing occurs locally on the device, without the need for cloud transfer. This approach offers advantages such as enhanced privacy, reduced energy consumption, and lower latency, making it particularly relevant for applications requiring real-time responses.

In this context, tools like LiteRT, formerly known as TensorFlow Lite, have enabled the execution of deep learning models on mobile and embedded devices. Studies have shown that it is possible to achieve significant reductions in model size and computational complexity, making it feasible to use on resource-limited devices (Diab and Rodriguez-Villegas, 2022). However, adapting these models to run efficiently in such constrained environments still presents challenges, particularly in optimizing inference speed and maintaining accuracy.

This work explores the development of an embedded device for human activity recognition, focusing on three specific movements: walking, standing still, and jumping. The goal is to demonstrate the feasibility of deploying machine learning models to operate efficiently on resource-constrained devices.

This paper is organized as follows. Section 2 presents related work on HAR and edge computing approaches. Section 3 describes the proposed system, including the hardware, data collection process, and model implementation. Section 4 discusses the experimental results, and finally, Section 5 provides conclusions and future directions.

[a] https://orcid.org/0009-0008-3190-3102
[b] https://orcid.org/0000-0001-5167-1523

Rodrigues, G. C. and Oliveira, R. A. R.
Human Activity Recognition on Embedded Devices: An Edge AI Approach.
DOI: 10.5220/0013475300003929
In *Proceedings of the 27th International Conference on Enterprise Information Systems (ICEIS 2025) - Volume 1*, pages 973-979
ISBN: 978-989-758-749-8; ISSN: 2184-4992

2 RELATED WORKS AND THEORETICAL REFERENCES

This section presents a literature and theoretical review on HAR, edge computing, and TensorFlow.

2.1 Human Activity Recognition (HAR)

Human activity recognition (HAR) has been extensively studied due to its applicability in areas such as healthcare and sports. Inertial sensor-based approaches (IMUs) have stood out for their ability to capture detailed human motion data, enabling accurate and real-time analysis. When combined with machine learning algorithms, these data allow human activities to be classified with high accuracy.

The study by (da Silva et al., 2023) developed a system using IMUs for the collection of motion data. The work evaluated three models of recurrent neural networks: long short-term memory (LSTM), gated recurrent unit (GRU) and simple RNN to classify human activities, demonstrating their effectiveness in recognizing motion patterns in HAR systems.

Likewise, (Waghchaware and Joshi, 2024) conducted a review of machine learning and deep learning techniques applied to human activity recognition. Their study explores the use of data from inertial sensors and images, highlighting the performance of different methods and providing a critical analysis of their advantages and limitations.

In another example, (Ann-Kathrin Schalkamp and Sandor, 2023) investigates the use of digital sensors for human action recognition to detect changes in movement patterns, aiming to identify diseases before clinical diagnosis. This approach highlights the potential of HAR systems not only for activity classification, but also for health monitoring and disease prevention. These studies underscore the relevance of HAR research, demonstrating its application in physical activities and health-related areas.

2.2 Edge Computing

Edge computing is an approach that processes data near the point of capture, reducing latency and reliance on remote connections. This technique is increasingly utilized in wearables and IoT devices, enabling real-time analysis, enhanced data privacy, and energy efficiency.

(M. S. Elbamby and Bennis, 2019) discusses the potential of edge computing for applications requiring high reliability and low latency, such as virtual reality and autonomous vehicles. The study highlights how local processing can enhance the scalability of wireless systems in critical scenarios.

Similarly, the work of (M. C. Silva and Oliveira, 2022) applied edge computing during the COVID-19 pandemic, presenting a proof of concept for an intelligent healthcare system. This system integrated wearable biometric sensors to monitor the vital signs of healthcare professionals, processing data locally to generate immediate insights. The research leveraged data fusion, big data, and machine learning techniques, demonstrating the benefits of this approach in health and safety contexts.

Meanwhile, (et al., 2024) highlight the crucial role of edge computing in human activity recognition, enabling devices to perform real-time inferences directly at the data capture point, without relying on cloud processing. This approach not only improves energy efficiency but also facilitates the development of autonomous devices capable of operating in environments with connectivity constraints.

2.3 TensorFlow and TensorFlow Lite

Machine learning (ML), a subfield of artificial intelligence, enables systems to learn patterns and make predictions based on data. Traditionally, ML models are trained and deployed in the cloud, with local devices like microcontrollers used only for data collection and transmission. However, the growing demand for real-time processing has led to frameworks like TensorFlow Lite, which allow models to run directly on resource-constrained devices.

TensorFlow Lite, now known as LiteRT, is designed to run machine learning models on microcontrollers and other hardware with severe memory and processing constraints (A. Haj-Ali and Weiser, 2020). It enables local inferences, eliminating the need for continuous cloud connectivity — a crucial factor for battery-operated devices. Additionally, reducing data transmission promotes energy efficiency. In Figure 1, the complete development flow using TensorFlow Lite is illustrated.

Recent studies, such as those by (S. S. Saha and Srivastava, 2022) and (et al., 2023), address techniques for optimizing models on low-capacity devices. These works explore methods like model compression, quantization, and strategies to maximize efficiency for specific tasks, such as signal classification.

Practical applications of TensorFlow Lite in human activity recognition have been presented by (V. Sharma and Cano, 2024) and (Bidyut Saha and Roy, 2024). These studies demonstrate the framework's effectiveness in microcontrollers for real-time

activity classification, highlighting TinyML as a viable solution for embedded devices in applications like health monitoring and intelligent wearable devices.

3 METHODS

In this section, we describe the steps and techniques involved in developing the embedded device for human activity recognition. Figure 1 illustrates the overall workflow used with TensorFlow Lite. The process is divided into three main stages: data collection, model training and conversion, and deployment on the device.

Figure 1: Development flow using TensorFlow Lite.

3.1 Data Collection

Movement data was collected using four wearable devices equipped with the BNO080 inertial sensor, which provides accelerometer, gyroscope, and magnetometer readings with 9 degrees of freedom. Each device was integrated with a NodeMCU ESP-32 microcontroller, powered by a rechargeable lithium-ion battery, ensuring portability and extended operation.

The devices were strategically positioned on different regions of the legs, as illustrated in Figure 2, to maximize movement capture and enhance activity recognition. Data transmission was carried out via Bluetooth at a frequency of 20 Hz, corresponding to one sample every 50 milliseconds, ensuring a high-resolution temporal dataset.

The dataset was collected from these four devices, each recording accelerometer and gyroscope data for three specific activities: walking, jumping, and standing still. Each sample comprised six normalized features (ax, ay, az, gx, gy, gz).

The recorded activities, as described in Table 1, include walking, standing still, and jumping. Data collection was conducted under various conditions to ensure the diversity and quality of the dataset. Raw data underwent preprocessing, including noise removal and normalization, ensuring compatibility with the machine learning model.

During data processing, each device recorded approximately 1,000 samples for each gesture. The collected data, sampled at a frequency of 20 Hz (1 sample every 50 ms), were grouped such that for each second of gesture duration, 20 samples were obtained. This procedure was applied individually to the data collected by each device.

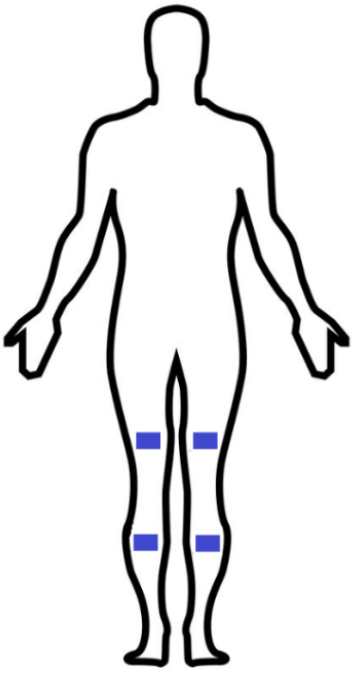

Figure 2: Positions of the wearable devices.

Table 1: Description of recorded activities.

Label	Description
Walking	Data collected indoors during a walking activity
Jumping	Data collected indoors during jumping activities
Standing	Data collected while standing still

Figures 3 and 4 display 250 samples of acceleration and gyroscope data for the activities of walking, standing still, and jumping. In the acceleration graph, walking shows significant variations across all three axes with typical cyclical oscillations. When standing still, the readings are stable and close to zero, with minor fluctuations. During jumping, the data exhibit peaks, especially on the y-axis, and more abrupt variations due to the intensity of the movement.

In the gyroscope graph, moderate variations in angular velocity during walking reflect the swinging of the legs, while at rest, angular velocity remains close to zero. During jumping, higher intensity peaks are observed but are less frequent than in the acceleration data.

3.2 Model Training and Conversion

In this stage, the first step was to implement a neural network model using the TensorFlow/Keras library to classify the actions of walking, standing still, and jumping based on six features per sample (ax, ay, az,

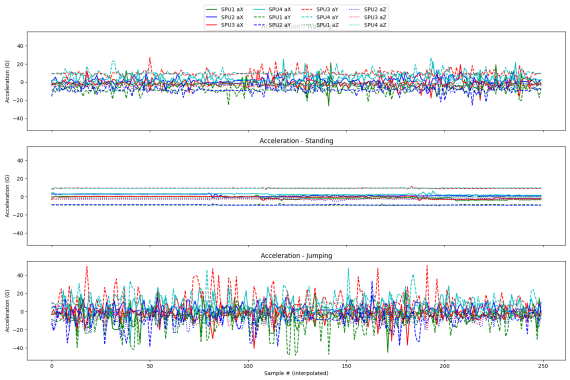

Figure 3: Collected acceleration data.

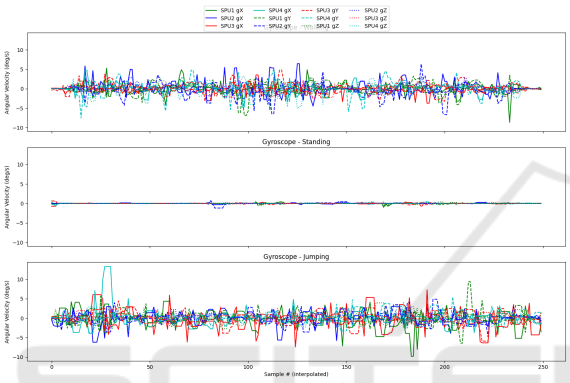

Figure 4: Collected gyroscope data.

gx, gy, gz). The model, shown in Figure 5, is defined as a sequential network composed of Dense Layers with ReLU activation, dropouts, and an output layer with Softmax activation.

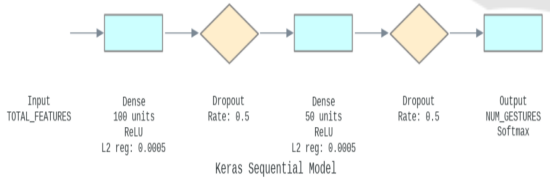

Figure 5: TensorFlow Model Structure.

The collected data was labeled using a one-hot encoding scheme, where each class was represented as a binary vector. The segmentation was performed by dividing the continuous sensor readings into non-overlapping windows of 20 samples per gesture, corresponding to 1 second of data per instance. This segmentation ensured that each instance captured a full second of motion data, preserving the temporal characteristics necessary for classification.

To ensure unbiased training, the dataset was randomly shuffled before splitting into 60% for training, 20% for validation, and 20% for testing.

The model was compiled using the Adam optimizer, an efficient choice for neural networks. The loss function employed was categorical_crossentropy. During training, the metrics monitored included accuracy and mean absolute error (MAE). The model was trained for up to 500 epochs with a batch size of 32, using the training and validation datasets.

Next, the model was evaluated to validate its performance with test data. Subsequently, it was converted into the TensorFlow Lite (TFLite) format for use in embedded devices. The model was then transformed into a byte array in C format using the xxd tool. The resulting data was stored in a header file named model.h, which can be directly integrated into the firmware of embedded systems. The final model had a size of 71316 bytes. Figure 6 illustrates the steps followed in this process.

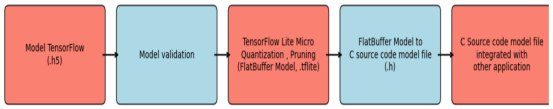

Figure 6: Model Training and Conversion Steps.

3.3 Deployment on the Device

During the deployment stage, the Arduino Nano 33 BLE board was used, as illustrated in Figure 7. This board was designed to cater to the growing audience of developers, makers, and enthusiasts in the fields of AIoT (Artificial Intelligence and Internet of Things), combining integrated sensors and Bluetooth Low Energy (BLE) connectivity. Equipped with an Arm® Cortex®-M4F processor (with FPU) operating at 64 MHz and 1 MB of Flash memory + 256 kB of RAM, the Arduino Nano 33 BLE is a complete and low-power board.

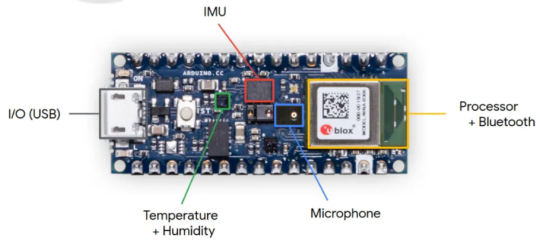

Figure 7: Arduino Nano 33 BLE.

To test the feasibility of the project, sensor data was emulated. The model was deployed on the board using the .h file generated in the previous stage. Using the emulated sensor features, such as acceleration on the X, Y, and Z axes, as well as gyroscope data on the same axes, the board was able to infer, locally and accurately, the type of movement performed, as shown in Figure 8.

```
walking: 0.013812
jumping: 0.986185
standing: 0.000003
Moviment detect: jumping
walking: 0.020038
jumping: 0.979962
standing: 0.000000
Moviment detect: jumping
walking: 0.691869
jumping: 0.307970
standing: 0.000161
Moviment detect: walking
walking: 0.194808
jumping: 0.805024
standing: 0.000168
Moviment detect: jumping
walking: 0.082878
jumping: 0.000974
standing: 0.916148
Moviment detect: standing
```

Figure 8: Experiment.

This experiment serves as a proof of concept and demonstrates the potential of the device to function as an edge system. It can receive sensor data and make decisions about the type of movement without needing to send the data to external servers. This enables more efficient operation, lower power consumption, and greater privacy—essential characteristics for the development of intelligent edge systems.

3.4 Evaluation Metrics

To assess the performance of the model and the embedded device, several metrics were used during both the training phase and testing with real data. Below, the main metrics are presented

3.4.1 During Training

- **Accuracy.** The ratio of correct predictions to the total number of samples evaluated.
- **Validation Loss (Val Loss).** Represents the average error of the model on the validation dataset. A low value indicates that the model is fitting the data correctly.
- **MAE (Mean Absolute Error).** The average of the absolute differences between predicted and actual values.

3.4.2 In the Test Data

- **Accuracy.** The percentage of correct predictions.

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \tag{1}$$

- **Precision.** The proportion of correctly predicted positive instances.

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{2}$$

- **Recall.** The proportion of actual positive instances correctly identified.

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{3}$$

- **F1-Score.** The harmonic mean of precision and recall.

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}} \tag{4}$$

- **Confusion Matrix.** A distribution of correct and incorrect predictions, highlighting potential misclassifications among classes.

3.4.3 In the Application

- **Inference Latency.** The time required to make a prediction from sensor data.
- **Power Consumption.** Measurement of energy consumption during inference.

$$\text{Power (mWs)} = \text{current (mA)} \cdot \text{voltage (V)} \cdot \text{time (s)} \tag{5}$$

4 RESULTS AND DISCUSSION

In this section, we present the results obtained during the development of the model and its deployment on the embedded device.

4.1 Model Performance During Training and Validation

Table 2 presents the metrics obtained during the training of the model, designed to recognize three human actions: walking, jumping and standing.

The results indicate that the model achieved good accuracy during training (93.65%) but a decline in validation accuracy (84.78%). This difference, along with the increase in loss and MAE, suggests early signs of overfitting. As a solution, it would be pertinent to investigate whether the dataset is sufficiently diverse or to apply training adjustments, such as regularization techniques or data augmentation.

Table 2: Metrics training.

Metric	Training	Validation
Accuracy	0.9365	0.8478
Val Loss	0.3340	0.5964
MAE	0.1120	0.1566

4.2 Evaluation on Test Data

Table 3 presents the global and per-class metrics based on the test data and the Figure 9 as confusion matrix, considering the classes: 0 as Walking, 1 as Jumping, and 2 as Standing.

Table 3: Metrics on test data.

Class	Accuracy	Precision	Recall	F1-Score
Walking	0.861	0.794	0.861	0.826
Jumping	0.631	0.857	0.631	0.727
Standing	0.971	0.918	0.971	0.944
Global	0.855	0.856	0.855	0.851

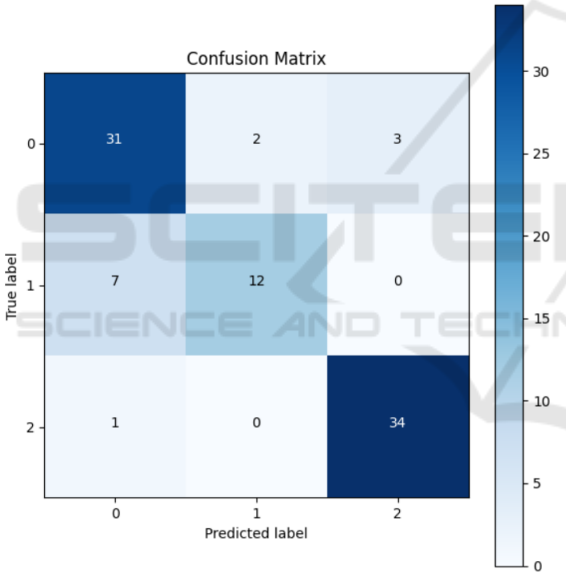

Figure 9: Confusion Matrix on Test Data.

The overall results indicate that the model achieved an accuracy of 85.56%, demonstrating its ability to correctly classify most samples. Furthermore, the global precision of 85.63% and global recall of 85.56% reflect a balance between the reliability of positive predictions and the correct identification of classes. The global F1-Score, which combines these metrics, was 85.15%, highlighting the model's overall consistency.

Analyzing the per-class results:

- Walking: This class showed good performance, with an accuracy of 86.11% and high recall, indicating that most examples were correctly classified. However, the moderate precision of 79.49% suggests some predictions for this class were misclassified as others.
- Jumping: This class showed the poorest performance, with an accuracy of 63.16% and low recall, indicating the model struggled to identify this class correctly. Despite this, the high precision (85.7%) suggests that the model made some correct predictions for this class but struggled to differentiate it from Walking.
- Standing Still: This class exhibited excellent performance across all metrics, with an accuracy of 97.14%, high recall, and high precision, demonstrating the model's reliability in identifying this class.

Although the model performs well overall, specific improvements can be made to increase accuracy and recall for the Jumping class, ensuring a more balanced classification across all categories.

4.3 Evaluation During Deployment on the Embedded Device

Table 4 presents the latency and energy consumption values during inference.

Table 4: Metrics during inference.

Metric	Result
Latency	3.2 ms
Energy Consumption	0.1056 mWs

These results indicate that executing the Machine Learning model developed with TensorFlow directly on the device (local inference) is highly efficient in terms of time and energy. The latency of only 3.2 ms demonstrates extremely fast processing, suitable for real-time applications such as motion detection. The energy consumption of 0.1056 mJ per inference is extremely low, reinforcing the feasibility of embedded device-based solutions for edge scenarios, such as wearables and IoT.

It is important to note that the latency of the system is also influenced by the sensor update rate. For instance, the sensor sampling rate, often measured in Hz, directly impacts the frequency at which data is acquired. A higher sampling rate could result in more frequent sensor readings, thus potentially increasing the data throughput and the need for faster processing times. Conversely, a lower sampling rate might reduce the load on the system but could compromise the real-time responsiveness of applications, such as motion detection, which require high-frequency data. Therefore, the choice of sensor update rate is a key

factor in achieving an optimal balance between performance and energy efficiency in embedded systems.

5 CONCLUSIONS

This study demonstrates the feasibility of developing a human activity recognition (HAR) model using TensorFlow and deploying it directly on an embedded device. The results highlight the practical application of machine learning at the edge, providing real-time performance and energy efficiency.

The process, which involved data collection, preprocessing, model training, and deployment, proved effective in recognizing three primary activities: walking, standing still, and jumping. The model achieved good accuracy during the training and validation phases, despite minor challenges such as overfitting and reduced performance in specific activity classifications, like jumping.

Deployment on the Arduino Nano 33 BLE validated the model's ability to perform accurate local inference with minimal latency and energy consumption. This makes it suitable for use in wearable devices and IoT systems, where privacy, efficiency, and low-power operation are crucial.

Future improvements could include diversifying the dataset to increase representativeness, optimizing the model's performance for classes with lower accuracy, and adding new activities to enhance the system's versatility. Additionally, in this study, sensor data on the Arduino Nano 33 BLE were emulated. As a next step, the device used for data collection could be integrated to send information directly to the Arduino Nano 33 BLE, eliminating emulation and creating a complete system.

Nevertheless, this proof of concept demonstrates the tremendous potential of embedded systems for real-time human activity recognition, highlighting the efficiency of edge AI solutions. The study establishes a solid foundation for future advances in HAR applications, meeting demands for privacy, low power consumption, and high efficiency.

ACKNOWLEDGEMENTS

The authors acknowledge the use of ChatGPT, an AI language model by OpenAI, for assisting with the translation of this article from Portuguese to English.

REFERENCES

A. Haj-Ali, N. P. and Weiser, U. (2020). Neural networks on microcontrollers: Saving memory at inference via operator reordering. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 1105–1117, Valencia, Spain.

Ann-Kathrin Schalkamp, Kathryn J. Peall, N. A. H. and Sandor, C. (2023). Wearable movement-tracking data identify parkinson's disease years before clinical diagnosis. *Nature Medicine*.

Bidyut Saha, Riya Samanta, S. G. and Roy, R. B. (2024). Towards sustainable personalized on-device human activity recognition with tinyml and cloud-enabled auto deployment. *arXiv preprint*, 2409.00093.

da Silva et al., J. C. F. (2023). Ai-based personalized human activity recognition in walking and trekking sports: A case study. *IEEE Latin America Transactions*, 21(8):874–881.

Diab, M. S. and Rodriguez-Villegas, E. (2022). Embedded machine learning using microcontrollers in wearable and ambulatory systems for health and care applications: A review. *IEEE Access*, 10:98450–98474.

et al., A. M. H. (2024). Tinyml empowered transfer learning on the edge. *IEEE Open Journal of the Communications Society*, 5:1656–1672.

et al., Y. A. (2023). A comprehensive survey on tinyml. *IEEE Access*, 11:96892–96922.

M. C. Silva, A. G. C. Bianchi, S. P. R. J. S. S. and Oliveira, R. A. R. (2022). Edge computing smart healthcare cooperative architecture for covid-19 medical facilities. *IEEE Latin America Transactions*, 20(10):2229–2235.

M. S. Elbamby, C. Perfecto, C.-F. L. J. P. S. S. X. C. and Bennis, M. (2019). Wireless edge computing with latency and reliability guarantees. *Proceedings of the IEEE*, 107(8):1717–1737.

S. S. Saha, S. S. S. and Srivastava, M. (2022). Machine learning for microcontroller-class hardware: A review. *IEEE Sensors Journal*, 22(22):21362–21390.

Shubham Gupta, Sweta Jain, B. R. and Deb, A. (2022). A tinyml approach to human activity recognition. *Journal of Physics: Conference Series*, 2273.

V. Sharma, D. P. and Cano, J. (2024). Efficient tiny machine learning for human activity recognition on low-power edge devices. pages 85–90.

Waghchaware, S. and Joshi, R. (2024). Machine learning and deep learning models for human activity recognition in security and surveillance: a review. *Knowledge and Information Systems*, 66:4405–4436.

APPENDIX

In this appendix, the source code for the wearable project can be found on GitHub: GitHub - Calça Wearable Project.