SHV Lab Assignment: Seated posture detection using an ESP32-CAM and Edgeimpulse

Noah Christian Le Roy, Noah Wijnheijmer

Master Applied Artificial Intelligence, *Amsterdam University of Applied Sciences*, Amsterdam, Netherlands
{noah.le.roy, noah.wijnheijmer}@hva.nl

***Abstract*—This research explores a low-cost approach to real-time posture monitoring by leveraging computer vision on an ESP32-CAM microcontroller paired with Edge Impulse for on-device processing. By developing a binary classifier, a proof-of-concept has been formulated that enables a hassle-free, locally-based posture monitoring system. The system can classify posture in real time without relying on cloud processing, ensuring low latency, affordability, and privacy. By leveraging edge Impulse, the model is efficiently quantized and optimized for edge computing. This process ensures that the model's quantization and optimization are efficiently optimized for the edge computing platform, enabling real time inference with minimal latency and power consumption. The findings will contribute to the development of cost-effective, posture monitoring solutions that can be easily deployed in everyday work environments.**

***Index Terms*—Posture Detection, Computer Vision, TinyML, Edge Computing, ESP32-CAM, Edge Impulse, Binary Classification, Embedded Systems**

I. Introduction

A. Research Topic:

The topic of this research is the integration of computer vision on Edge devices for real-time posture detection. The goal of this research is to explore a promising alternative to conventional posture monitoring solutions. By leveraging lightweight AI models, efficient hardware, and embedded processing, this approach aims to offer a cost-effective and privacy-preserving method for monitoring and improving posture without reliance on cloud-based computation.

B. Research Problem:

Sitting in the wrong posture for long periods has been shown to cause back and neck pain [1], lower productivity [2], and long-term health risks [3]. With a large part of the population working behind desks for extended hours, there's a growing need for practical and affordable posture monitoring solutions [4].

Current systems, like wearable sensors can be intrusive and uncomfortable, which often results in low user compliance [5], Non-sensor approaches (e.g., manual observation or self-reporting) often rely on subjective judgments and demand active user involvement, leading to inconsistent or infrequent monitoring [6]–[8]. They lack real-time feedback; users may remain in a poor posture until the next manual check, potentially worsening musculoskeletal strain [6]. These limitations underscore why automated sensor-based methods—providing continuous, objective tracking—are frequently preferred [7], [8].

a TinyML Edge device using a camera provides a simple nearly full-body posture analysis by capturing visual context like shoulder posture [9].

Despite advances in computer vision and AI, there's little research on affordable, easy-to-use solutions that can track sitting posture in real-time without needing cloud connectivity [10]. The Smart Health and Vitaly lab of the HvA has expressed interest in a solution for this using the ESP-32-CAM. The system formulated in this paper contributes to filling this gap.

C. Research Purpose:

This research accumulates into the development and evaluation of a proof-of-concept binary classification system for detecting poor sitting posture using computer vision on an ESP32-CAM microcontroller. By leveraging Edge Impulse for on-device AI inference, the system can classify posture in real-time without relying on cloud processing, ensuring low latency, affordability, and privacy.

D. Research Question:

Based on the research topic, problem, and purpose, the following research question has been formulated:

What is the potential of computer vision leveraging Edge Impulse on a TinyML device in enabling an efficient, affordable solution for real-time posture detection?

E. Research Structure:

This paper is organized into four main sections, followed by an Appendix:

- **Introduction (Section I)**: Presents the research topic, problem, and overall purpose, culminating in the key research question.
- **Background (Section II)**: Reviews the theoretical framework and relevant technologies, including TinyML, the ESP32-CAM, and Edge Impulse.
- **Model (Section III)**: Details the study's methodology, hardware setup, data collection process, and model training/deployment steps.

- **Discussion (Section IV)**: Interprets the results, addresses limitations, and suggests potential improvements to enhance model robustness and user experience.

The **Appendix** provides step-by-step instructions (user manual) on configuring the Arduino IDE, capturing images, and deploying the trained model on the ESP32-CAM.

II. Theoretical Framework

A. *Micro controllers*

A microcontroller (MCU) is a compact integrated circuit designed for embedded systems, combining a processor, memory, and input/output peripherals on a single chip. Unlike general-purpose computers, microcontrollers are optimized for specific tasks, making them ideal for automation and control applications. They process data in real time, interact with sensors and actuators, and support various communication protocols. MCUs are widely used in consumer electronics, industrial automation, robotics, and IoT devices due to their low power consumption, affordability, and efficiency. Popular microcontrollers include Arduino-based ATmega chips, ESP32, and STM32 series, each tailored for different performance and connectivity needs.

B. *ESP32-CAM*

The ESP32-CAM [11] is a small, low-cost development board that integrates an ESP32 microcontroller with a camera module, enabling image and video processing capabilities for edge computing applications. It is based on the ESP32-S chip, which includes Wi-Fi and Bluetooth connectivity, making it ideal for IoT projects.

Key Features of ESP32-CAM:

- ESP32-S Processor: A dual-core 32-bit Xtensa LX6 CPU with clock speeds up to 240 MHz.
- OV2640 Camera Module: A 2-megapixel camera capable of capturing images and streaming video.
- Wireless Connectivity: Built-in Wi-Fi and Bluetooth for remote access and control.
- Compact Form Factor: Small PCB size (27x40 mm) without USB interface.
- MicroSD Card Slot: Supports up to 4GB for local image/video storage.
- Low Power Consumption: Suitable for battery-powered applications.

C. *TinyML*

TinyML is the deployment of machine learning models on ultra-low-power devices like microcontrollers, enabling local data processing without cloud dependency. This reduces latency, enhances privacy, and allows AI to function in offline environments. Due to the limited memory and processing power of microcontrollers, models are optimized using techniques like quantization and pruning.

TinyML is widely used in predictive maintenance, healthcare, agriculture, and consumer electronics. It enables real-time AI applications such as anomaly detection in industrial machines, health tracking in wearables, and voice recognition in smart devices. These systems operate on minimal power, often running for years on small batteries.

Frameworks like Edge Impulse and TensorFlow Lite for Microcontrollers help developers train and deploy models on hardware like the Arduino Nano 33 BLE Sense, Raspberry Pi Pico, and ESP32.

D. *Edge Impulse*

Edge Impulse [12] is a platform designed to facilitate the development, training, and deployment of machine learning models on edge devices, such as microcontrollers, sensors, and embedded systems. It provides an end-to-end workflow for building AI models that process sensor data in real time while consuming minimal power.

The platform allows users to collect data from various sources, including accelerometers, microphones, and cameras, and preprocess it using built-in signal processing tools. It supports TinyML (tiny machine learning) techniques, enabling AI to run efficiently on low-power hardware. Developers can train models using classical machine learning or deep learning techniques and optimize them for deployment on edge devices.

Edge Impulse offers integrations with popular hardware platforms like Arduino. This intigration streamlines testing and deployment of the model. It also supports edge deployment through its optimized SDKs.

A key advantage of Edge Impulse is its ability to bring AI capabilities to resource-constrained devices (like the ESP32). This reduces reliance on cloud computing, lowers latency, and enhances data privacy by processing information locally on the device.

E. *Computer Vision*

Computer vision is a field of artificial intelligence that enables computers to analyze and interpret visual data, allowing them to recognize objects, detect patterns, and extract meaningful information from images and videos. Object classification, a fundamental task in computer vision, has evolved from classical techniques based on handcrafted features to modern deep learning approaches.

Classical object classifiers rely on manually designed feature extraction techniques, such as edge detection, texture analysis, and shape descriptors. Algorithms like SIFT, HOG, and ORB extract these features, which are then fed into machine learning models such as Support Vector Machines or Random Forests for classification. These methods work well for specific tasks but struggle with complex, high-variance data due to their limited ability to generalize across different conditions.

Modern object classification methods, particularly those based on deep learning, have largely replaced classical approaches. Convolutional Neural Networks (CNNs) and newer architectures like Vision Transformers (ViTs) automatically learn and extract hierarchical features from raw image data, eliminating the need for manual feature engineering. While deep learning models achieve superior accuracy and robustness, they require large datasets and high computational power, making them less feasible for resource-constrained environments where classical methods may still be relevant. However, smaller, less resource-intensive deep learning models, such as TinyML-based classifiers, can still be deployed on edge devices, enabling real-time object classification in low-power, constrained environments. These models offer a balance between efficiency and accuracy, making them suitable for applications where cloud computing is not feasible.

III. Model

A. Methodology

In this section, we briefly describe the practical setup for using the ESP32-CAM and capturing posture data. For more detailed, step-by-step instructions (including IDE setup, camera configuration, and data acquisition), see our *User Manual* in Appendix A.

B. Hardware

The following components have been used for this project:

- **ESP32-CAM** [11]
- **ESP32-CAM Programmer Shield** [13]
- **Micro USB Cable**
- **Small amount of PLA filament** for the enclosure

To ensure the ESP32-CAM is well-protected and seamlessly integrated into the final product, we adapted an existing 3D model found online to fit the specific layout of our module and its components. This enclosure will serve multiple purposes: shielding the module from dust and accidental damage, providing structural support, and enhancing the device's overall appearance. The design will incorporate essential features such as ventilation for heat dissipation, easy access to necessary components, modifications were made on the initial design mainly to accommodate the lens placement. The final 3D model was printed using a Bambu P1S and a small amount of black PLA filament.

C. Data Collection

To train a binary classifier that can distinguish between good and suboptimal sitting positions, a dataset has been collected.

This has been done in a controlled indoor environment with consistent lighting, background, and camera placement. Furthermore, all images are captured using the ESP32-CAM, ensuring consistency in resolution between the training and deployment phases. A relatively small amount of images is required, as the model leverages the Edge Impulse platform to apply transfer learning, enabling it to build on pre-trained features and achieve efficient learning with limited data.

The dataset consists of three classes, following the same methodology as the image classification tutorial from Edge Impulse [14]. Having three classes enables us to also include unknown in the classification, helping account for uncertainty. The classes are:

- **Good Sitting Position** – Images of a person sitting with proper posture, labeled as "good sitting position."
- **Suboptimal Sitting Position** – Images of a person sitting with poor posture, labeled as "bad sitting position."
- **Unrelated** – Images that do not depict a person sitting (e.g., empty chair, no person detected), labeled as "unrelated."

Figure 1 shows representative images from each class to illustrate the variety of training examples. All images were captured using the ESP32-CAM's onboard camera, preserving consistency in quality, resolution, and perspective between training and deployment. Consistent data collection helps prevent discrepancies between training and real-world inference, improving the model's accuracy.

All pictures have been taken in a controlled environment using the same lighting condition and the same test subject. This choice was largely driven by the current hardware constraints, as the setup lacks the compute capacity for extensive generalization across diverse environments. Moreover, the non-wide-angle camera limits the observable field of view, which can be suboptimal for full-body posture analysis.

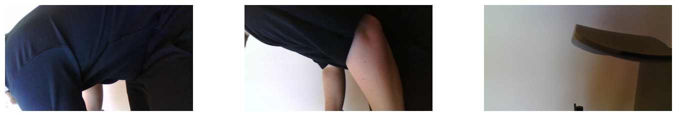


Fig. 1. Examples of the three classes in the dataset: (Left) *Suboptimal Sitting Position*, (Center) *Good Sitting Position*, (Right) *empty*.

D. Considerations and Limitations

Because all the training data is taken in the same environment, under the same lighting conditions and angle using the same test subject, results will likely be hard to replicate in different settings. Eventhough not being optimal for real world scenarios, this is a necessity for creating a working proof of concept.

Furthermore, it is important to acknowledge that classifying a sitting position as simply "good" or "suboptimal" is not always straightforward. Posture can be context-dependent, varying between individuals based on factors such as body type, flexibility, and medical conditions. Some positions that may appear "suboptimal" in a general sense could be neutral or even necessary for certain individuals. This initial scope of this research is towards a general binary classification on sitting positions, but there might be benefits in further research towards training and testing on individual postures, treating anomalies as changes with respect to the normal sitting pattern of that individual [5].

For this study, the classification will follow widely accepted ergonomic guidelines for sitting posture. However, this binary distinction simplifies a more complex issue, and further refinement may be required to account for individual variability and edge cases.

E. Model Training

To train a classifier, Edge Impulse serves as our machine learning development platform. The labeled images are uploaded to Edge Impulse, where optimizations is performed within its graphical user interface (GUI). These optimizations will focus on adjusting hyperparameters. The hyperparameters for this project are as follows: TRAINING CYCLES: 20 LEARNING RATE: 0,0005 Data augmentation in this setup is enabled by selecting the "Data augmentation" checkbox in Edge Impulse's Transfer Learning menu. Once checked, Edge Impulse automatically flips the training images (and may apply other slight transformations), creating virtual variations of the same data. This increases the model's robustness without requiring additional manual data collection. The built-in loss function used by Edge Impulse for classification models is Categorical Cross-Entropy [15]. As shown in Figure 2, the model learned steadily over time.

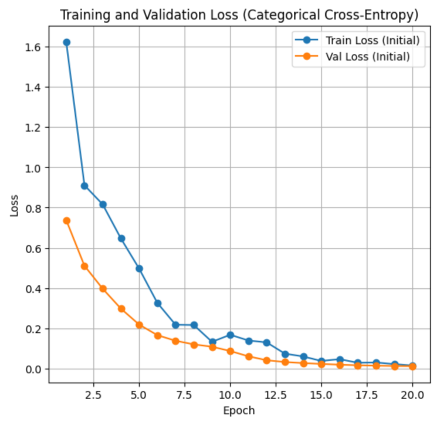


Fig. 2. The plot shows that both training and validation loss steadily decrease over 20 epochs, indicating that the model is learning effectively.

F. Model Deployment

Using the built-in transfer learning options on Edge Impulse [16], our data has been fine-tuned on the MobileNetV2 [17] architecture.

It is then deployed using Edge Impulse, which optimizes and compiles the model into firmware specifically designed for the ESP32-CAM. This process ensures that the model is efficiently quantized and optimized for edge computing, enabling real-time inference on the ESP32-CAM with minimal latency and power consumption. The firmware is flashed onto the ESP32-CAM, allowing it to perform on-device image classification without requiring a constant connection to a cloud server.

The actual classification results can be viewed through the Arduino Serial Monitor. All real-time interaction with the embedded model happens via the Arduino IDE and its Serial Monitor. Edge Impulse handles data collection and model creation (exporting the model as an Arduino library), after which the ESP32-CAM is programmed in the IDE to execute that model.

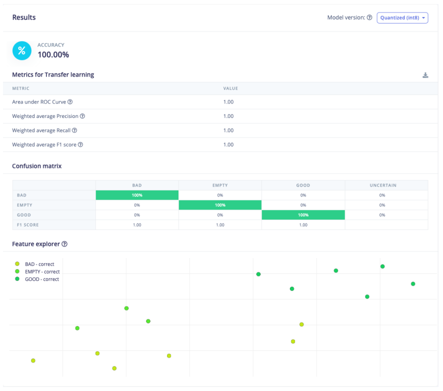
G. Implementation

1) **Hardware Preparation:** Follow the wiring and board settings as explained in the User manual (Appendix A.)
2) **Software/IDE Setup:** Ensure that the Arduino IDE and ESP32-CAM libraries are properly installed (see instructions in the manual).
3) **Data Collection:** Use the onboard camera streaming interface to capture various sitting postures.

During real-world trials, the posture classification model generally works well but shows a slight lag in detecting when the user switches from "good" to "bad" posture. Additionally, it is quite sensitive to changes in the environment (e.g., lighting, camera angle), and requires the user's posture to be similar to the positions in the training data. If the setting or posture deviates significantly from what the model has seen, its accuracy may decrease.

H. Results

Figure 3 presents the final evaluation metrics and confusion matrix as reported by Edge Impulse. All classes (`BAD`, `EMPTY`, and `GOOD`) were correctly classified, yielding accuracy, precision, recall, and F1-scores of 1.00. This perfect performance reflects the controlled nature of data collection and a small dataset—primarily following Edge Impulse's minimal images-per-class recommendation. Consequently, the model has not been exposed to diverse lighting conditions, camera angles, or posture variations, raising concerns about overfitting and limited robustness. While it performs well in scenarios that closely match its training setup, additional data or stronger regularization may be needed to ensure reliable performance in less controlled, real-world environments.



Results

ACCURACY
100.00%

Metrics for Transfer learning

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	1.00
Weighted average Recall	1.00
Weighted average F1 score	1.00

Confusion matrix

	BAD	EMPTY	GOOD	UNCERTAIN
BAD	100%	0%	0%	0%
EMPTY	0%	100%	0%	0%
GOOD	0%	0%	100%	0%
F1 SCORE	1.00	1.00	1.00	

Feature explorer

Fig. 3. Edge Impulse test results showing 100% accuracy and perfect confusion matrix.

I. Conclusion

This study successfully demonstrates a low-cost posture detection approach using an ESP32-CAM microcontroller and Edge Impulse. By collecting a small, controlled dataset and applying transfer learning on a pre-trained model, the system accurately classifies three categories: *good posture*, *suboptimal posture*, and *unrelated*. Crucially, all training and inference tasks run directly on the device, eliminating the need for continuous cloud connectivity. Although the limited dataset and controlled data-collection environment constrain immediate real-world applicability, this project lays a solid foundation for potential extensions to more diverse scenarios and additional user feedback mechanisms. Overall, it confirms the feasibility of embedded computer vision solutions for real-time posture monitoring in resource-constrained settings using an image classification model.

IV. Discussion

This work represents a proof of concept in a controlled environment, aiming primarily to show that a low-cost microcontroller like the ESP32-CAM can run a TinyML model for real-time posture detection. Because we relied on a small dataset captured under uniform lighting and with a single test subject, the model is not guaranteed to generalize well to diverse or real-world conditions.

A more sophisticated approach would incorporate pose estimation—mapping key points such as shoulders, elbows, and hips, and calculating angles to determine whether the user is seated correctly. Technologies such as `ultralytics/pose` [18] could serve as a starting point for that, though the hardware requirements may exceed the current ESP32-CAM setup.

User feedback mechanisms also warrant improvement. Presently, classification output is only accessible via the Arduino Serial Monitor; implementing an on-device display, beeping alarm, or push notification would offer a more immediate and user-friendly warning of suboptimal posture. Furthermore, the ESP32-CAM's limited field of view makes it challenging to capture the entire upper body; replacing it with a wide-angle camera could improve posture analysis for more active or variable work settings.

Ultimately, further research is needed to validate these methods in different environments with more diverse datasets—encompassing varied lighting, seating, and user physiques. This additional work would strengthen the system's ability to maintain high accuracy outside the controlled conditions demonstrated here, helping ensure it can effectively serve a broader population with meaningful, real-time posture insights.

Appendix

A. IDE Setup

To install and configure the Arduino IDE, follow these steps:

1) Download the Arduino IDE from https://www.arduino.cc/en/software/.
2) Install the ESP32 board definitions in the Board Manager.
3) Select `AI Thinker ESP32-CAM` as your target board.

B. Data Collection

Use the `CameraWebServer` example to capture images directly from the ESP32-CAM. Adjust the resolution as needed, and organize your images by posture class (e.g., "Good", "Bad", "Empty") before uploading them to Edge Impulse.

C. Deployment

After training your model in Edge Impulse and exporting as an Arduino library, add the .zip library in the Arduino IDE. Load the provided example sketch, select the correct board, and flash to the ESP32-CAM.

For a complete guide on replicating this setup step by step—including firmware flashing, Edge Impulse configuration, and data collection—please refer to our dedicated user manual.

References

[1] E. Szczygieł, K. Zielonka, S. Metel, and J. Golec, "Musculo-skeletal and pulmonary effects of sitting position – a systematic review," 2017. [Online]. Available: https://www.aaem.pl/pdf-72599-9828?filename=9828.pdf

[2] H. Daneshmandi, A. Choobineh, H. Ghaem, and M. Karimi, "Adverse effects of prolonged sitting behavior on the general health of office workers," 2017. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC5618737/

[3] W. Gao, T.-Y. Lin, Y.-C. Chen, C.-Y. Hsu, and C. P. Wen, "Occupational sitting time, leisure physical activity, and all-cause and cardiovascular disease mortality," 2024. [Online]. Available: https://jamanetwork.com/journals/jamanetworkopen/fullarticle/2814094

[4] G. V. Research, "Posture correction market size, share, growth report, 2030," 2024. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/posture-correction-market-report

[5] P. Vermander, A. Mancisidor, I. Cabanes, and N. Perez, "Intelligent systems for sitting posture monitoring and anomaly detection: an overview," *Journal of NeuroEngineering and Rehabilitation*, vol. 21, no. 28, 2024. [Online]. Available: https://jneuroengrehab.biomedcentral.com/articles/10.1186/s12984-024-01322-z

[6] M. G. Silva, B. M. Pilling, and C. T. Candotti, "Body posture self-assessment tools: a scoping review," *Fisioterapia e Pesquisa*, vol. 30, p. e22017823, 2023.

[7] P. Vermander, A. Mancisidor, I. Cabanes, and N. Perez, "Intelligent systems for sitting posture monitoring and anomaly detection: an overview," *Journal of NeuroEngineering and Rehabilitation*, vol. 21, p. 28, 2024.

[8] S. Sen, V. González Sánchez, E. J. Husom, S. Tverdal, S. Tokas, and S. O. Tjøsvoll, "ERG-AI: enhancing occupational ergonomics with uncertainty-aware ML and LLM feedback," *Applied Intelligence*, vol. 54, no. 23, pp. 12 128–12 155, 2024.

[9] U. Lee, S. Lee, S.-A. Kim, J.-D. Lee, and S. Lee, "Validity and reliability of the single camera markerless motion capture system using rgb-d sensor to measure shoulder range-of-motion: A protocol for systematic review and meta-analysis," *Medicine*, vol. 102, no. 22, p. e33893, 2023.

[10] G. Martino, M. Grasso, G. P. Ingrassia, A. Sorrentino, and F. Tramontana, "A deep learning approach for human posture classification using imu sensors," *Applied Sciences*, vol. 14, no. 18, p. 8557, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/18/8557

[11] TinyTronics, "Esp32-cam wifi and bluetooth board with ov2640 camera," 2025, accessed: 13-Feb-2025. [Online]. Available: https://www.tinytronics.nl/en/development-boards/microcontroller-boards/with-wi-fi/esp32-cam-wifi-and-bluetooth-board-with-ov2640-camera

[12] I. Edge Impulse, "Edge impulse: The leading development platform for machine learning on edge devices," 2025, accessed: 13-Feb-2025. [Online]. Available: https://www.edgeimpulse.com/

[13] TinyTronics, "Esp32-cam-mb programmer shield," 2025, accessed: 13-Feb-2025. [Online]. Available: https://www.tinytronics.nl/en/development-boards/accessories/adapter-boards/esp32-cam-mb-programmer-shield

[14] E. Impulse, "Image classification - edge impulse documentation," 2024, accessed: 2025-03-26. [Online]. Available: https://docs.edgeimpulse.com/docs/tutorials/end-to-end-tutorials/computer-vision/image-classification

[15] Edge Impulse, "Loss functions," 2024, accessed: 2025-04-9. [Online]. Available: https://docs.edgeimpulse.com/docs/concepts/machine-learning/neural-networks/loss-functions#customizing-loss-function-in-expert-mode

[16] ——, “Transfer learning (images),” https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/transfer-learning-images, 2025, accessed: 2025-04-11.

[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” pp. 4510–4520, 2018. [Online]. Available: https://arxiv.org/abs/1801.04381

[18] Ultralytics, “Ultralytics pose estimation,” https://docs.ultralytics.com/tasks/pose/, 2024, accessed: 2025-04-11.