Posture Detection and Assistance for Mobility Aid: A Sensor Fusion and Machine Learning Approach

Ivan Omeko, Stefan Twieg and Martin Obert
Department of Electrical, Mechanical and Industrial Engineering, Anhalt University of Applied Sciences, Bernburger Str. 55, 06366 Köthen, Germany
I.Omeko@outlook.com, Stefan.Twieg@hs-anhalt.de, Martin.Obert@hs-anhalt.de

Keywords: K-Means, CNN, Neural Networks, Posture Detection, Elderly Care, Machine Learning, Jetson Orin, MQTT.

Abstract: The increasing proportion of elderly individuals poses significant challenges for mobility assistance and independent living. Rollators are widely used mobility aids, yet conventional devices remain largely passive and provide no feedback on improper usage that may lead to discomfort or musculoskeletal strain. This paper presents the design and evaluation of a smart posture detection system integrated into a commercially available rollator as part of the AktiMuW project. The proposed system combines multiple sensors - including inertial measurement units, ultrasonic distance sensors, and strain gauges - with machine learning techniques to assess rollator usage and user posture in real time. A two-stage classification approach is employed. First, the operational state of the rollator is identified using supervised learning methods. Feed-forward neural networks, convolutional neural networks, and random forest classifiers are evaluated, with the random forest model demonstrating the best balance of accuracy and computational efficiency, achieving over 97% validation accuracy across all device states while significantly reducing inference time and resource usage. Second, user posture is analyzed using unsupervised k-means clustering. Different posture granularities are investigated, ranging from five detailed posture classes to simplified configurations. A three-class posture model ("Comfortable," "Too Close," and "Too Far") is selected as an optimal compromise between classification accuracy and actionable feedback, achieving validation accuracies of up to 99%. The complete system is deployed on an embedded NVIDIA Jetson Orin Nano platform and integrated via MQTT-based communication. Real-time benchmarking confirms that the combined models operate within acceptable computational limits while maintaining reliable posture detection. The presented approach demonstrates the feasibility of lightweight, sensor-based posture monitoring for rollator users and provides a foundation for future assistive feedback systems aimed at improving safety and comfort for elderly individuals.

1 INTRODUCTION

1.1 Motivation

People around the world are living longer due to improvements in quality of life and healthcare. In Germany, the share of people aged 65+ reached 23% in 2024 - 2 percentage points higher than ten years earlier - and by 2050, this figure is expected to rise to 28% [1].

With age, physical abilities decline, and walking often becomes more challenging. Among individuals over the age of 65, 66.5% of those with a disability report significant walking difficulties [2].

To address these public health and technological challenges, many older adults rely on mobility aids such as rollators. Rollators (wheeled walkers) are a primary mobility aid for the elderly, used by a lot of older adults in Germany. However, traditional rollators are passive and limited in functionality.

The AktiMuW project [3] was initiated to bridge this gap by integrating smart technologies into rollators. The project aims to implement assistive features such as obstacle and hazard detection, navigation support, and posture detection to enhance user safety and independence.

There were previous approaches to track posture with walkers, which involved complex setups [4] or relied on additional sensors like cameras [5]. To the authors' knowledge, the approach of combining motion, weight-bearing force, and the person's distance using sensor fusion and machine learning to detect posture in a rollator, which will be discussed later, represents a novelty.

1.2 Problem Statement

When elderly individuals use rollators incorrectly, it can place unnecessary stress on parts of the spine or certain muscle groups, particularly in the back, leading to discomfort or even injury. Ensuring proper posture while using a rollator is essential for maintaining comfort and preventing fatigue.

This paper describes the development of a posture detection system for rollator users. The system is based on data collected from various sensors, including distance measuring sensors, inertial measurement units (IMUs), and strain gauges. These inputs are used to train and deploy machine learning models capable of assessing user posture in real-time and providing corrective feedback.

2 SYSTEM DESCRIPTION

The AktiMuW project adopts a strategy of incorporating smart technologies like voice integration [6] or detection of daily objects in urban traffic into a commercially available rollator without altering its structural integrity. A combination of sensors was selected specifically for posture analysis, including inertial measurement unit (IMU), distance measurement sensors, and strain gauges. Recognizing the complexity and variability of human movement, a machine learning-based approach was selected for interpreting the gathered sensor data. This methodology represents the foundational step toward enabling real-time posture detection. The process of the posture detection is shown in Figure 1.

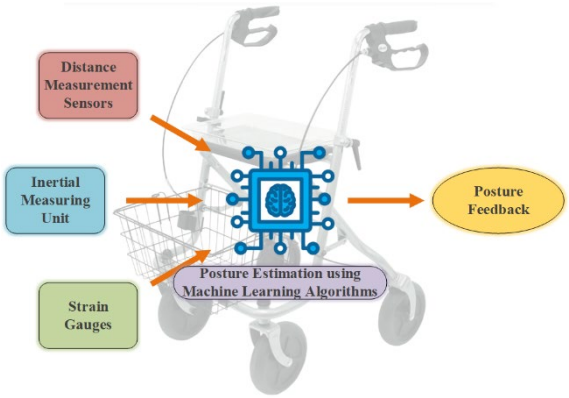

Figure 1: Visualisation of the posture detection in the AktiMuW project.

2.1 Hardware

The project was based upon the Drive Migo 2G rollator. The smart system integration required additional hardware components to enable sensor data collection and processing capabilities. The selected hardware architecture consists of:

- Jetson Orin Nano Developer Kit for machine learning processing;
- Raspberry Pi Zero W for sensor data acquisition;
- MPU6050 IMU that measures rollator acceleration and angular velocity for movement detection;
- HC-SR04-P ultrasonic sensors to monitor proximity between rollator and user's legs for positioning analysis;
- strain gauges for measuring the load applied to rollator handles indicating user grip and weight distribution;
- HX711 amplifiers for amplification and digitization of the strain gauge signals for accurate load measurements.

Raspberry Pi with IMU, amplifiers and other complimentary electronics have been placed on a custom in-house manufactured printed circuit board (PCB). Strain gauges have been mounted to each of the rollator's handles using clear silicone adhesive sealant.

To keep the electronic hardware safe, a custom housing was developed using SolidWorks 2023 and 3D-printed to accommodate the Jetson and the PCB. The housing was placed under the seat of the rollator. Two HC-SR04-P sensors have been attached to the housing via mounting part. Hardware placement is shown in Figure 2.

Additionally, some of the sensors were calibrated prior to use. The MPU6050 IMU's output was adjusted for its static tilt offset. The selection of the primary distance sensor was based on a comparative test of the HC-SR04, HC-SR04-P, and TFmini-S models; the full evaluation is detailed in a separate study [7].

2.2 Software

The Raspberry Pi Zero W serves as the primary data acquisition unit, responsible for sensor data collection and transmission to the Jetson Nano for processing and analysis. The system implements a structured data flow from sensor collection through network transmission to final processing.

Figure 2: Housing with components fitted onto the rollator.

Data transmission utilizes the MQTT (Message Queuing Telemetry Transport) protocol, a lightweight messaging standard optimized for IoT applications with limited bandwidth and processing requirements. The Jetson Nano plays the role of the MQTT broker while simultaneously hosting a dedicated wireless access point. The Raspberry Pi connects to this network as an MQTT client, establishing a reliable communication channel between the data acquisition and processing units.

Sensor data from utilized sensors is merged into a unified data string format to ensure consistent parsing and processing. Individual sensor readings are delimited by semicolons, creating a structured message that maintains data integrity during transmission. All sensor data is published under the standardized MQTT topic "Rollator" to enable organized message routing and subscription management. Upon reception at the MQTT broker, incoming messages are processed according to the predefined data structure. Each message contains sensor readings in the following sequence:

- ultrasonic sensor distance measurements (left and right sensors);
- IMU linear acceleration data (x, y, z axes);
- IMU angular velocity data (x, y, z axes);
- strain gauge load measurements (left and right sensors);
- environmental temperature reading (acquired via IMU sensor).

2.3 Posture Detection Methodology

Accurate posture classification depends on the reliable identification of the rollator's operational state. Therefore, a two-stage classification approach was implemented, where the system first determines the current state of the device before proceeding to analyse the user's posture.

Three distinct device states were defined to represent the primary usage conditions of the rollator system:

- State 0: Device is not used and not moved;
- State 1: Device is used but not moved (user applies pressure on the handles but without movement);
- State 2: Device is used and moved.

This classification provides essential context for subsequent posture detection since posture assessment is only meaningful when the device is actively used and moved (the system is in State 2).

In the context of the posture detection, additional classification classes were defined:

- State 3: Comfortable;
- State 4: Too low, middle;
- State 5: Too low, close;
- State 6: Too high, middle;
- State 7: Too high, far.

The only adjustable parameter on the rollator related to the user's comfort is the height of the handles. Improper handle height can cause the user to adopt compensatory postures: handles positioned too low may lead to excessive forward leaning, while handles set too high may result in backward leaning. However, users may also modify their posture by changing their distance from the rollator. For instance, leaning may be reduced by stepping closer to the rollator when handles are too low or stepping farther away when handles are too high, but the device usage will remain incorrect.

Table 1: Defined postures and their descriptions.

Posture	Description
Comfortable	Handles at 13 cm (hands at pelvic height), user's legs approx. 45 cm from ultrasonic sensors
Too low, middle	Handles at 18 cm on the bar, user's legs approx. 45 cm from ultrasonic sensors
Too low, close	Handles at 18 cm on the bar, user's legs approx. 15-20 cm from ultrasonic sensors
Too high, middle	Handles at 8 cm on the bar, user's legs approx. 45 cm from ultrasonic sensors
Too high, far	Handles at 8 cm on the bar, user's legs approx. 60 cm from ultrasonic sensors

Detailed descriptions of the postures are provided in Table 1. Here, "middle" refers to the distance between the user and the rollator associated with the comfortable posture, while "close" and "far" describe compensatory adjustments in user distance in response to low or high handle positions, respectively. Since the testing was performed by the

author, every configuration was determined according to personal ergonomic preferences.

3 CHOSEN MACHINE LEARNING APPLICATION

For the purpose of the device state classification (States 0-2) the following approaches have been chosen for comparison: feed-forward neural networks (FNN), convolutional neural networks (CNN) and random forests (RF).

FNNs are one of the most fundamental and widely used types of neural networks. In FNNs, information flows in one direction: from input to output, without cycles or loops. During the learning process, FNNs adjust their weights using algorithms such as gradient descent and Widrow-Hoff's learning rule, which minimize the prediction error. A classic example is the perceptron: a simple FNN used for binary classification tasks like face recognition. FNNs are particularly effective for pattern recognition and function approximation tasks due to their ability to generalize. Once trained on a dataset, the network can produce accurate predictions for previously unseen inputs that share structural patterns with the training data [8].

CNNs were specifically designed to address the limitations of FNNs in dealing with high-dimensional data such as images. While they share the same conceptual foundation - layers of neurons with learnable weights - The defining operation in a CNN is convolution. In this process, small filters (also known as kernels) slide over the input image, extracting local patterns such as edges, textures, or shapes [9].

A random forest is an ensemble-based classification algorithm that constructs multiple decision trees and aggregates their predictions through majority voting. Each tree in the forest is trained on a bootstrap sample - a randomly drawn subset of the training data with replacement - and, during training, each split in a tree considers only a random subset of features rather than the full feature set. This dual randomness through both data and feature selection, encourages diversity among trees and reduces overfitting, a common problem with individual decision trees [10].

For the task of the posture detection, the k-means clustering algorithm was selected. In contrast to the previous approaches, k-means is an unsupervised learning method. While supervised models are trained using labeled data (the model knows the target class during training), unsupervised methods like k-means operate on unlabeled data, grouping data points based solely on inherent patterns and similarities.

K-means is a partitional clustering algorithm that organizes a dataset of n samples into k clusters by minimizing the within-cluster variance. Each cluster is represented by a centroid, and the goal of the algorithm is to assign data points to the cluster whose centroid is nearest, using a distance metric such as Euclidean distance [11]. The algorithm proceeds iteratively through two main steps:

1) assignment step: each data point is assigned to the cluster with the closest centroid;
2) update step: the centroids are recalculated as the mean of all data points assigned to each cluster.

These steps repeat until the cluster assignments no longer change significantly, typically indicating convergence [12].

4 MODELS TRAINING

4.1 Device State Classification

To perform the model training, data was collected for each class individually and saved to separate CSV files using a dedicated Python script. Later they were combined into a single dataset.

For the device state classification such features were selected:

- Linear acceleration along the X-axis (Acc_x);
- Linear acceleration along the Y-axis (Acc_y);
- Angular velocity around the Z-axis (Gyro_x);
- Strain gauge readings from both handles (DMS_R, DMS_L).

FNN, CNN and RF models were trained on the same dataset with the same dataset parameters. Results of the conducted trainings are presented in the cross-validation confusion matrixes below. The validation results in Figure 3 are: State 0: 99.03%, State 1: 91.50%, State 2: 94.52%. This indicates that the FNN trained model can detect the correct state of the rollator in the vast majority of cases.

The CNN in Figure 4 produced validation results similar to the FNN, with a modest increase in performance: approximately 5 and 0.3 percentage points higher accuracy at detecting States 1 and 2 respectively.

Among the three classification methods, all achieved validation accuracies exceeding 90%. The RFC in Figure 5 demonstrated the highest overall performance, with validation accuracies of 99.76% for State 0, 98.61% for State 1 and 97.81% for State 2.

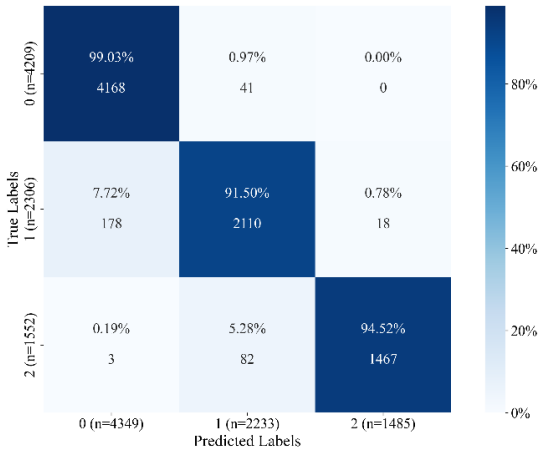

Figure 3: Cross-validation confusion matrix of the trained FNN model.

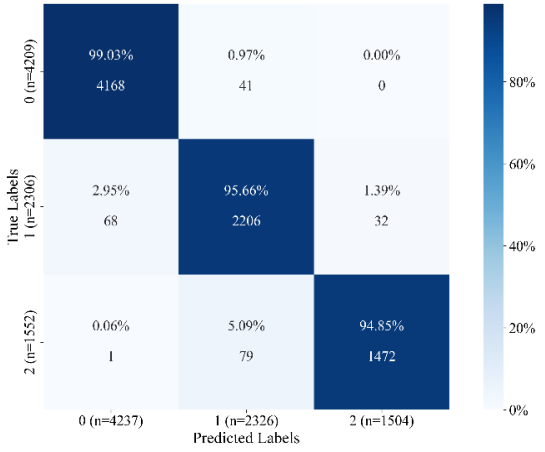

Figure 4: Cross-validation confusion matrix of the trained CNN model.

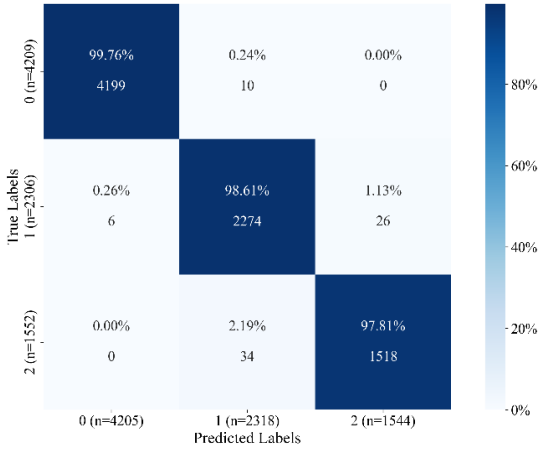

Figure 5: Cross-validation confusion matrix of the trained random forests model.

To evaluate the real-time performance of the trained models and choose the best performing one, dedicated Python scripts were written for each classifier. These scripts executed the models on the Jetson Orin Nano for 300 seconds (5 minutes) under controlled conditions - no additional processes were active beyond essential system services. During the test run, the following performance metrics were recorded:

- Inference time for each prediction;
- RAM usage by the process during inference;
- System-wide CPU usage at 1-second intervals.

As can be observed from Table 2, both FNN and CNN models show similar performances with CNN being slightly more resource efficient, using on average less RAM and CPU. The RFC model, on the other hand, showed a significant advantage in speed, with an average inference time of 38.16 ms, compared to 114.75 ms for FNN and 115.20 ms for CNN. This translates to roughly 67% faster inference compared to both FNN and CNN. Random forests model performs utilizing system resources in much more efficient way: on average it uses around 150 MB of RAM, which is 73% less than both FNN and CNN.

On average, RFC consumed only 5.28% of total CPU load, while FNN and CNN both hovered around 14%: RFC used around 63% less CPU than the neural networks. This lower CPU usage is especially important for maintaining overall system responsiveness on the Jetson Orin Nano.

Even under peak conditions (top 1% and top 0.1% of samples), RFC consistently maintained lower CPU usage: 8.90% with RFC vs. ~18% with FNN/CNN for top 1% CPU usage, 9.30% with RFC vs. ~19% with FNN/CNN for top 0.1% CPU usage.

Overall, the RFC model is the most resource-efficient option across all metrics: it has fastest inference time, lowest RAM consumption and lowest average and peak CPU usage. For this reason, the RFC-trained model was chosen to be used for real-time deployment on the Jetson Orin Nano due to its lightweight and efficient operation.

4.2 K-Means Clustering for Posture Detection

Data collection for posture classification was conducted using a procedure similar to that employed for the initial three-class device state classification. Individual datasets corresponding to each defined posture were recorded separately and save to CSV files. These datasets were subsequently merged into a single training set during the clustering process.

Table 2: Real-time performance metrics of device state classification models.

Metric	FNN	CNN	RFC
Total number of inferences	2100	2042	2030
Average inference time	114.75 ms	115.20 ms	38.16 ms
Average RAM usage	559.04 MB	555.19 MB	153.30 MB
Average CPU usage	14.40%	14.12%	5.28%
Average Top 1% CPU usage	18.36%	18.20%	8.90%
Average Top 0.1% CPU usage	19.20%	18.70%	9.30%

The k-means clustering approach was applied to the dataset in an unsupervised manner using the trained sensor data. Since clustering does not inherently assign semantic meaning to its output, the resulting cluster labels (0 through 4) do not correspond directly to the actual posture classes.

To overcome this, majority voting was employed: each cluster was analyzed to determine which actual posture label appeared most frequently within it. These dominant labels were then mapped to the corresponding cluster indices using a cluster_to_label dictionary, which was saved to disk for future use.

Once training was completed, the trained model was used to predict cluster assignments for the validation set. The predictions were then translated into meaningful posture classes via the mapping dictionary.

Clustering with earlier defined five postures was performed. Cluster formation is shown in Figure 6, using features from ultrasonic sensors and strain gauges. The cross-validation was conducted as well.

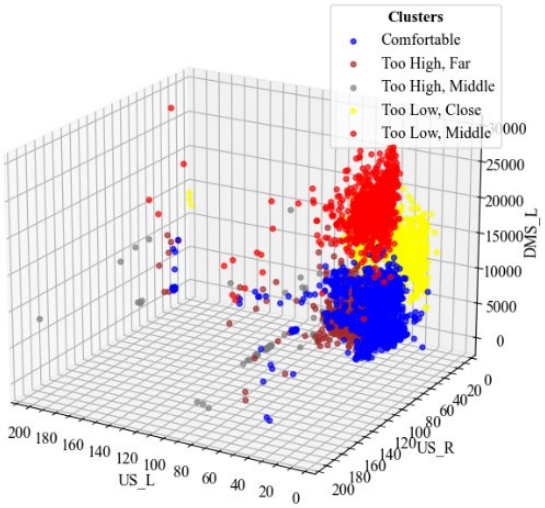

Figure 6: Graphical representation of 5 posture cluster using US_L, US_R and DMS_L.

The confusion matrix (Fig. 7) shows mixed results. Clustering struggled to identify certain postures correctly. Although “Comfortable”, “Too High, Far”, “Too Low, Close” and “Too Low, Middle” were predicted correctly 81% or more, "Too High, Middle" posture was consistently misclassified, most often as "Comfortable" (75%). This suggests overlap in feature space between the two postures, particularly in distance measurements. Although "Too High, Middle" is defined by increased handle load, the difference in strain gauge data appears insufficiently distinct to separate it from the "Comfortable" posture during unsupervised learning.

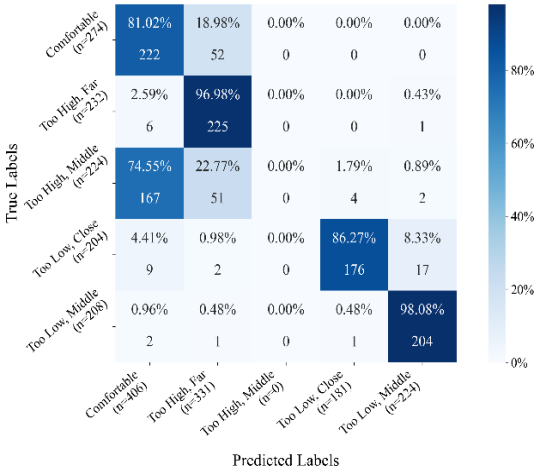

Figure 6: Cross-validation confusion matrix of the performed 5-posture clustering.

To assess whether reducing the number of clusters could improve clarity and accuracy, a second trial was conducted using just two clusters:

- Comfortable;
- Not Comfortable.

“Comfortable” posture remains the same as it was in the first trial. The dataset for “Not Comfortable” includes the data for every specific uncomfortable posture out of four that was established, combined into one. Cluster visualization and the resulting confusion matrix are presented in Figure 8 and Figure 9 respectively. This reduced-class setup showed very high classification accuracy:

"Comfortable" - 86%, "Not Comfortable" - 98%. While these results demonstrate improved performance, a binary distinction between postures is too coarse for the needs of the project. More informative feedback is necessary.

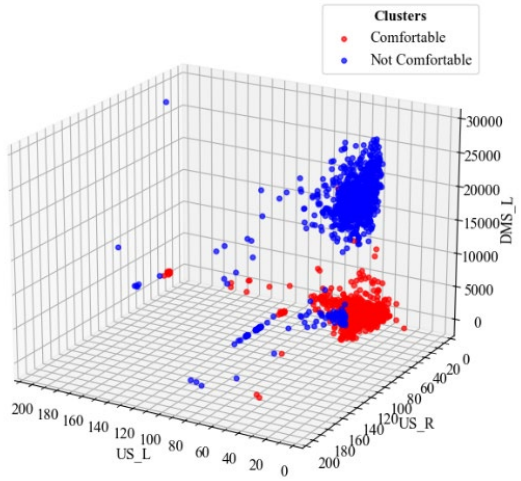

Figure 7: Graphical representation of 2 posture clusters using US_L, US_R and DMS_L.

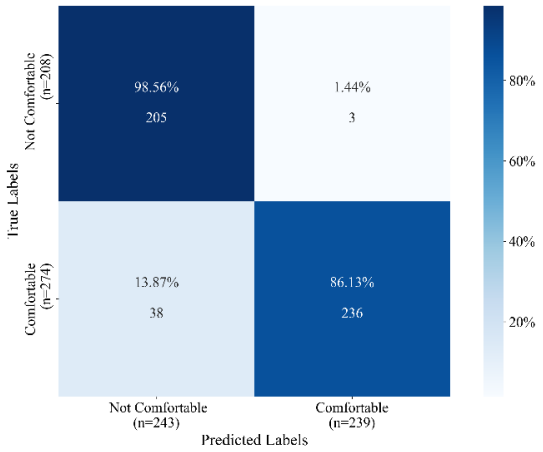

Figure 8: Cross-validation confusion matrix of the performed 2-posture clustering.

For that reason, a third trial was conducted with three postures:

- Comfortable;
- Too Close - Comfortable handle height (13 cm) and short distance (15-20 cm);
- Too Far - Comfortable handle height (13 cm) and long distance (60 cm).

This configuration provides clear, interpretable posture feedback to the user while remaining manageable for the clustering algorithm. Clusters are shown in Figure 10 and validation results in Figure 11.

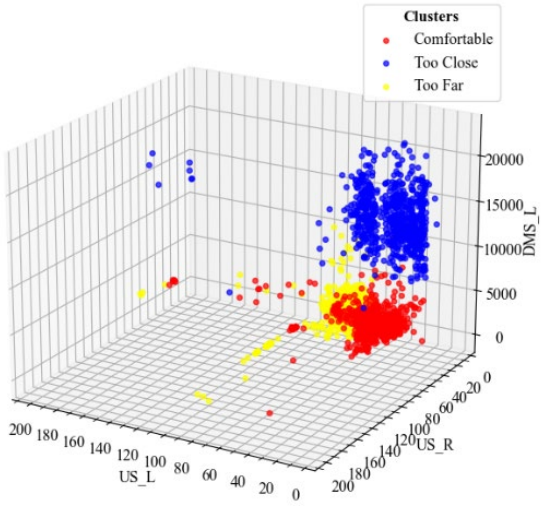

Figure 9: Graphical representation of 3 posture clusters using US_L, US_R and DMS_L.

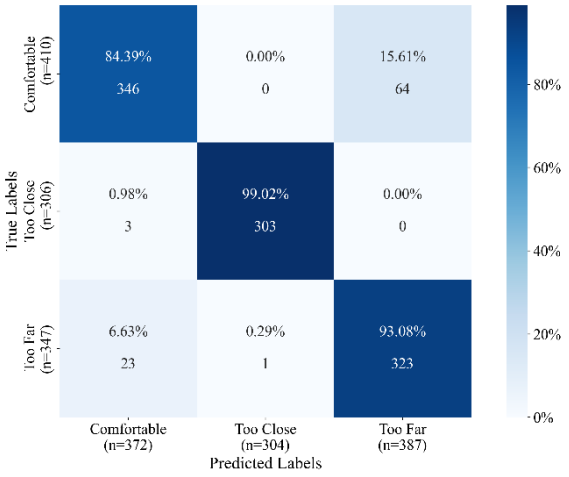

Figure 10: Cross-validation confusion matrix of the performed 3-posture clustering.

Cross-validation confusion matrix shows high accuracy at predicting 3 postures. "Comfortable" posture was correctly predicted 84% of the time, "Too Close" - 99% and "Too Far" - 93%.

The results indicate that while five-class clustering provides detailed classification it struggles with accuracy due to overlapping features among similar postures. Reducing to two clusters improves accuracy but sacrifices granularity. The three-cluster setup stands as a strong compromise, maintaining high classification accuracy while offering meaningful posture differentiation.

This balanced approach enables the system to deliver meaningful, actionable posture feedback in real-time without adding unnecessary computational complexity. For these reasons, the three-posture clustering model was chosen for deployment on the rollator, running on the Jetson Orin Nano.

5 FINAL SETUP

With the models for device state classification and posture detection trained, they were integrated into a unified deployment script for the rollator system. A Python script was developed that uses the random forest classifier to classify the device state (labels 0, 1, or 2). When State 2 is detected, indicating walking, the script triggers the k-means clustering model to identify the user's posture. All necessary components, including the machine learning models and scalers, were installed on the Jetson Orin Nano within the ~/posture_detection/ directory. A dedicated virtual environment was created to ensure safe and isolated execution.

To evaluate the system's resource usage when both models run concurrently, a benchmarking test was conducted, similar to that described in Section 4.1. The results are summarized in Table 3.

Table 3: Real-time performance metrics: RFC only vs. RFC + k-means.

Metric	RFC	RFC + k-means
Total number of inferences	2030	2017
Average inference time	38.16 ms	41.08 ms
Average RAM usage	153.30 MB	155.12 MB
Average CPU usage	5.28%	10.72%
Average Top 1% CPU usage	8.90%	15.75%
Average Top 0.1% CPU usage	9.30%	16.30%

The results show that while the system experiences a moderate increase in load when both models are active, resource usage remains within acceptable limits. The slight increases in inference time (from 38.16 ms to 41.08 ms) and RAM usage (from 153 MB to 155 MB) are minimal. Although average CPU usage doubled, peak usage remained below 17%, indicating that the script operates efficiently and leaves system headroom.

During live posture estimation, predicted posture labels are published to the MQTT topic Rollator/Labeled (Fig. 12), enabling external systems or interfaces to access real-time feedback. The integrated posture detection setup was tested extensively and consistently performed well, accurately identifying postures in the majority of cases. Minor deviations were occasionally observed, likely due to sensor interference or transient signal noise.

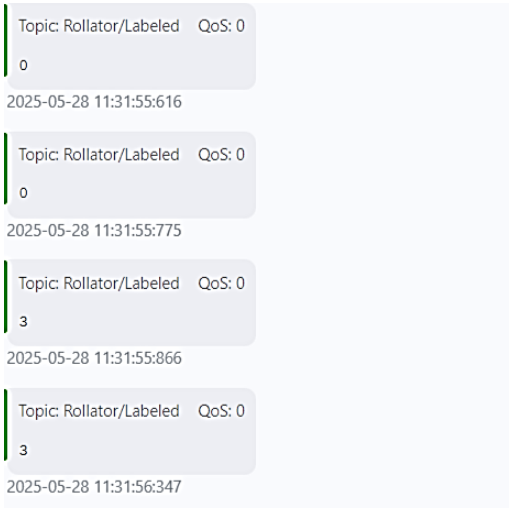

Figure 11: Posture estimated labels published to the MQTT "Rollator/Labeled" topic.

6 CONCLUSIONS

For device state classification (States 0, 1 and 2) feed-forward network, convolutional neural network and random forest approaches have been compared. The RFC model outperformed others in both accuracy and resource efficiency.

For posture detection, k-means clustering approach was used to differentiate between five postures, but for the accuracy it was decided to use the three-posture configuration with "Comfortable", "Too Close" and "Too Far" postures. Finally, both the RFC and k-means trained models were successfully integrated and deployed on the Jetson Orin Nano for the real-time posture detection.

The underlying approach demonstrates that real-time tracking of the body posture while using a rollator is successfully achievable through the use of multiple sensors and their fusion via machine learning.

To further advance the goal of preventing falls and postural damage, future iterations will incorporate a broader dataset to achieve the highest possible level of user compatibility.

ACKNOWLEDGEMENTS

The research project underlying this contribution was funded by the Federal Ministry of Education and Research under the grant number FKZ 03WIR3122A. The responsibility for the content of this publication lies with the author.

REFERENCES

[1] Destatis, "Bevölkerungspyramide: Altersstruktur Deutschlands von 1950 - 2070," [Online]. Available: https://service.destatis.de/bevoelkerungspyramide/index.html#!y=2024&a=20,65&l=en&g.

[2] W. He and L. J. Larsen, Older Americans With a Disability: 2008-2012, Washington, DC: U.S. Government Printing Office, 2014.

[3] Hochschule Anhalt, "AktiMuW," [Online]. Available: https://www.hs-anhalt.de/aktimuw/uebersicht.html.

[4] R. Pérez-Rodríguez, P. Moreno-Sánchez, M. Valdés-Aragonés, M. Oviedo Briones, S. Divan, N. García-Grossocordón, and L. Rodríguez-Mañas, "FriWalk robotic walker: usability, acceptance and UX evaluation after a pilot study in a real environment," Disability and Rehabilitation: Assistive Technology, vol. 15, pp. 1-10, 2019, [Online]. Available: https://doi.org/10.1080/17483107.2019.1617795.

[5] M. Palermo, J. M. Lopes, J. André, A. C. Matias, J. Cerqueira, and C. P. Santos, "A multi-camera and multimodal dataset for posture and gait analysis," Scientific Data, vol. 9, no. 1, 2022, [Online]. Available: https://doi.org/10.1038/s41597-022-01722-7.

[6] S. Rajanayagam, M. A. Ingrisch, P. Müller, P. Jahn, and S. Twieg, "Enhancing Voice Activity Detection for an Elderly-Centric Self-Learning Conversational Robot Partner in Noisy Environments," Proceedings of the International Conference on Applied Innovations in IT, vol. 13, 2025, [Online]. Available: https://doi.org/10.25673/119209.

[7] I. Omeko, S. Twieg, and M. Obert, "Sensor-Based Gait Analysis: A Comparative Study of Ultrasonic and Laser Sensors for Gait Monitoring in Rollator-Assisted Walking," Proceedings of the International Conference on Applied Innovations in IT, vol. 13, pp. 213-222, 2025, [Online]. Available: https://doi.org/10.25673/119236.

[8] I. Aleksander and H. Morton, An Introduction to Neural Computing, London: Chapman and Hall, 1990.

[9] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[10] L. Breiman, "Random Forests," Machine Learning, vol. 45, pp. 5-32, 2001, [Online]. Available: https://doi.org/10.1023/A:1010933404324.

[11] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern Classification, 2nd ed., Wiley, 2006.

[12] R. Nascimento, A. Neto, Y. Shalom, H. Nascimento, L. Lucena, and J. Freitas, "A new hybrid optimization approach using PSO, Nelder-Mead Simplex and Kmeans clustering algorithms for 1D Full Waveform Inversion," PLOS ONE, vol. 17, 2022, [Online]. Available: https://doi.org/10.1371/journal.pone.0277900.