

Reducing computational cost for real-time fall detection: A Mahalanobis pre-filtering approach

Luong Cong Duan*, Nguyen Ngoc Minh, Truong Cao Dung, Tran Thi Thuc Linh

Faculty of Electronics Engineering 1 & EDA Lab, Posts and Telecommunications Institute of Technology, 96A Tran Phu, Ha Dong, Hanoi, Vietnam.

*Corresponding author: duanlc@ptit.edu.vn

Received 25 Feb. 2026; Revised 14 Jun. 2026; Accepted 15 Sep. 2026; Published 15 Oct. 2026.

DOI: <https://doi.org/10.54939/1859-1043.j.mst.114.2026.47-54>

ABSTRACT

Real-time fall detection using wearable devices is essential for TinyML applications, but continuous neural network inference can be expensive for low-power microcontrollers. This paper presents a two-stage fall detection framework that utilizes Mahalanobis distance as a lightweight pre-filter before a CNN classifier. In the first stage, accelerometer data is converted into a nine-dimensional statistical feature vector and compared against a reference distribution of non-fall movements. Windows that fall below a specified threshold are considered low-risk and bypass the CNN, while suspicious ones proceed to the second stage for classification. Evaluated on the KFall dataset with 62,204 windows, the Mahalanobis pre-filter achieved fall recall rates of 98.18% and 98.55%, and false-positive rates of 39.35% and 37.40%, respectively. On an ESP32-S3 microcontroller, the first stage required only 196 microseconds per window, resulting in an average computation reduction of about 58.9-60.6% compared to CNN-only inference.

Keywords: Fall detection; Mahalanobis distance; TinyML; Wearable sensors; Computational efficiency.

1. INTRODUCTION

The global demographic shift toward an aging population has increased the demand for reliable fall detection systems to ensure the safety of elderly individuals [1]. Wearable devices equipped with Inertial Measurement Units (IMUs) are popular for their non-intrusive nature and continuous monitoring of physical activity [2]. However, implementing advanced detection algorithms on resource-constrained microcontroller units (MCUs) presents challenges related to computational overhead and energy consumption [3, 4]. Existing wearable systems often struggle to balance high detection accuracy with constraints on battery life and processing power [5].

Early fall detection systems typically used simple threshold rules based on acceleration magnitude or body angle [6]. While these methods are fast and easy to implement, they can generate many false alarms during dynamic activities like jogging or lying down. Recent advances involve deep learning models, such as convolutional neural networks (CNNs) [7, 8], recurrent neural networks (RNNs), and long short-term memory (LSTM) networks, which can learn temporal patterns from sensor data [9, 10]. More broadly, human activity recognition systems also depend on well-defined temporal segments, as shown in frame-level video annotation for dynamic Vietnamese sign-language gestures [11]. However, continuous inference with these models is demanding for small, embedded platforms, resulting in a trade-off between detection accuracy and deployment cost [12].

Some studies have focused on optimizing edge-based fall detection using model compression techniques such as weight pruning and quantization, which reduce memory and latency but often require extensive retraining and can degrade detection sensitivity [8, 13]. Other researchers have proposed multi-stage systems that initially utilize simple magnitude-based thresholds [14], but these approaches frequently fail to capture the complex correlations among accelerometer axes [15]. Threshold-based preprocessing has also been used in radar-based human activity classification to retain meaningful micro-Doppler regions before CNN classification, improving

accuracy over unprocessed inputs [16]. The Mahalanobis-based approach presented here addresses these issues by employing a multivariate statistical metric that maintains high discriminative power with minimal computational overhead [17].

This study presents a two-stage framework for low-cost fall detection on wearable embedded devices. In the first stage, the Mahalanobis distance is used to synthesize nine statistical features into a single distance score, which helps determine whether to filter a window or send it to the CNN classifier. The approach leverages the combined behavior of acceleration axes and employs a straightforward decision logic with a single adjustable threshold. A lower threshold enhances sensitivity, while a higher threshold decreases CNN activation for better computational efficiency. The experimental protocol encompasses pre-fall and post-fall transition windows, utilizes a subject-level split, and reports MCU timing on the ESP32-S3 hardware. The paper is structured as follows: Section II outlines the two-stage framework and decision logic; Section III details the experimental setup and metrics; Section IV discusses pre-filtering performance; and Section V concludes with future research directions.

2. PROPOSED METHODOLOGY

2.1. System architecture

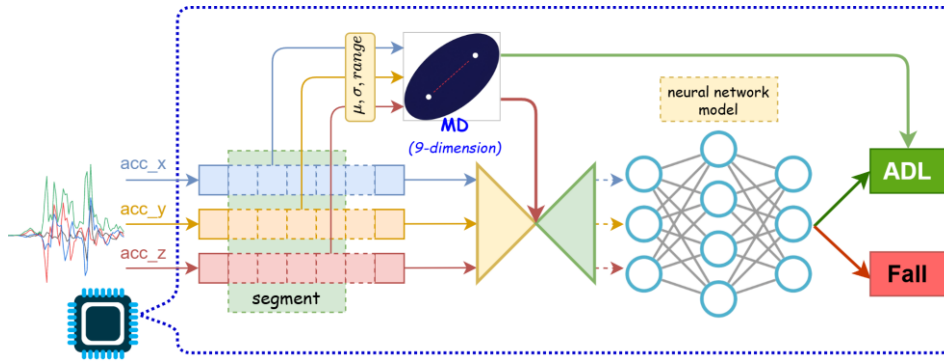


Figure 1. Proposed two-stage fall detection framework using a Mahalanobis pre-filter and CNN.

The proposed system features a two-stage structure for real-time fall detection on wearable devices. The first stage uses a Mahalanobis distance pre-filter to evaluate accelerometer signal windows. If the distance from a non-fall reference distribution falls below a threshold, the window is labeled low-risk and bypasses the second stage. If it exceeds the threshold, the window is processed by a CNN classifier. This design allows the Mahalanobis stage to select candidates for the CNN while filtering out many normal activities, minimizing unnecessary CNN activations. While a false positive in the first stage only triggers an extra CNN check, a false negative is critical as it prevents the CNN from assessing potentially dangerous movements. By focusing expensive CNN computations on complex patterns, the system efficiently maintains high detection reliability.

2.2. Stage 1: Mahalanobis distance pre-filtering

In Stage 1, statistical features are extracted from each accelerometer window to represent physical movement patterns. For each data segment, a nine-dimensional feature vector f is generated as follows:

$$f = [\mu_x, \sigma_x, r_x, \mu_y, \sigma_y, r_y, \mu_z, \sigma_z, r_z] \quad (1)$$

where μ_i , σ_i and r_i denote the mean, standard deviation, and range (maximum - minimum) of the accelerometer signal for axis $i \in x, y, z$, respectively.

The Mahalanobis distance, D_M , is then calculated to measure the similarity between the current

feature vector and a reference distribution representing routine activities. The following equation defines the distance:

$$D_M(f) = \sqrt{(f - \mu)^T S^{-1} (f - \mu)} \quad (2)$$

In this expression, $\mu \in \mathbb{R}^9$ represents the mean feature vector and $S \in \mathbb{R}^{9 \times 9}$ is the covariance matrix estimated from training samples of non-fall activities. To ensure real-time performance on resource-constrained hardware, the inverse covariance matrix S^{-1} is pre-computed offline and stored in the device memory. Finally, the computed D_M value serves as the primary metric for the pre-filtering decision logic. The system compares D_M against a predefined threshold τ . If $D_M < \tau$, the movement is classified as a routine activity and filtered out immediately without invoking the neural network. This stage effectively separates common daily motions from complex or sudden events that require further analysis in the second stage.

2.3. Stage 2: Neural network

Table 1. CNN architecture used as the second-stage classifier.

Layer	Output shape	Parameters
Input	(None, 100, 4)	0
Slicing and reshape	(None, 100, 1, 1)	0
Channel convolutions (x4)	(None, 10, 1, 16)	2,176
Concatenate	(None, 40, 1, 16)	0
Reshape	(None, 4, 10, 16)	0
Conv2D	(None, 1, 8, 32)	6,176
Flatten	(None, 256)	0
Dense	(None, 64)	16,448
Dense	(None, 32)	2,080
Output dense	(None, 2)	66
Total		26,946

The second stage of the framework is activated when the computed Mahalanobis distance exceeds the predefined threshold τ . Upon triggering, the system retrieves raw tri-axial accelerometer data from a circular buffer synchronized with the trigger for detailed classification. In contrast to the statistical summaries used in the pre-filtering stage, the neural network processes full temporal sequences, enabling better distinction between actual falls and dynamic daily activities.

This stage can use different neural network architectures, such as CNNs, RNNs, or LSTM networks. In this study, we use a compact CNN model implemented in our previous work as the second-stage classifier. The CNN input is a 2-s window with 100 samples and 4 channels: acc_x , acc_y , acc_z , and RMS acceleration. The network uses separate channel convolutions, concatenation of feature maps, a 2-D convolution layer, and fully connected layers for binary classification. It contains 26,946 trainable parameters and requires approximately 174K FLOPs per inference. In the proposed framework, this CNN is executed only for candidate windows selected by the Mahalanobis pre-filter.

3. EXPERIMENTAL

3.1. Dataset

The proposed framework's performance is evaluated using the KFall dataset [18], which includes movement recordings from 32 healthy participants. Data were collected using an IMU sensor at the waist, sampled at 100 Hz, and downsampled to 50 Hz for efficiency. The continuous data were segmented into 2-second windows with 50% overlap, resulting in 62,204 samples. This includes 35,889 windows for ADLs, 24,471 for pre- and post-fall situations, and 1,844 for specific

fall events. While useful for controlled evaluation, the KFall dataset lacks representation of the target elderly population, as the subjects are young and healthy. Therefore, these results serve only as an initial validation for the proposed embedded method, highlighting the need for further testing with older adults in daily-life conditions.

3.2. Train-test protocol

The dataset was divided into two groups, Group A and Group B, with each group containing all windows from 16 randomly selected participants. This subject-level division prevented overlapping windows from the same subject from appearing in both the training and test sets. Two evaluation runs were conducted: Group A was used as the training set and Group B as the test set in the first run, and the roles were reversed in the second run. Mahalanobis reference parameters, including the mean vector and covariance matrix of the non-fall training windows, were estimated only from the training group.

After obtaining the Mahalanobis distances for the training group, the threshold τ was selected solely from the training data. The threshold selection prioritized fall safety, requiring fall recall in the training set to be at least 99%. Among the thresholds that met this criterion, the one that provided better non-fall filtering was selected. The estimated Mahalanobis parameters and the selected threshold were then fixed and applied to the unseen test group for evaluation. In this experiment, the selected thresholds were $\tau = 2.13$ for the Group A training run and $\tau = 2.07$ for the Group B training run.

3.3. MCU deployment

The proposed two-stage framework was implemented on an ESP32-S3 MCU running at 240 MHz. The firmware was compiled using *xtensa-esp32s3-elf-gcc*. The deployment was designed to measure the runtime cost of both processing stages under the same embedded platform. For Stage 1, the measured processing path includes min-max normalization, extraction of the 9 statistical features, and computation of the Mahalanobis distance. The inverse covariance matrix and the selected threshold were stored in device memory, so the online computation only required feature calculation and matrix-based distance evaluation.

For Stage 2, the CNN classifier was deployed after quantization to reduce memory usage and inference cost on the MCU. The quantized CNN was executed only for windows that passed the Mahalanobis pre-filter. The execution time of both Stage 1 and Stage 2 was then measured on the ESP32-S3 to evaluate the practical computational cost of the proposed framework.

3.4. Evaluation metrics

The first-stage pre-filter is evaluated as a candidate selector. The positive class means that a window is forwarded to the CNN. The negative class means that the window is filtered out.

Recall is used as the main safety metric:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

where TP is the number of fall windows forwarded to the CNN, and FN is the number of fall windows filtered out by the pre-filter.

The false positive rate (FPR) measures how many non-fall windows still activate the CNN:

$$FPR = \frac{FP}{FP + TN} \quad (4)$$

Specificity measures the filtering ability for non-fall windows:

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (5)$$

The activation rate is the percentage of all windows sent to the CNN:

$$\text{Activation rate} = \frac{TP + FP}{TP + FP + TN + FN} \quad (6)$$

The filtering rate is the percentage of all windows bypassing the CNN:

$$\text{Filtering rate} = \frac{TN + FN}{TP + FP + TN + FN} \quad (7)$$

For computational analysis, the CNN-only baseline runs the CNN for every window. The proposed hybrid system runs the Mahalanobis pre-filter for every window and runs the CNN only for activated windows.

4. RESULTS AND DISCUSSION

4.1. Mahalanobis pre-filtering performance

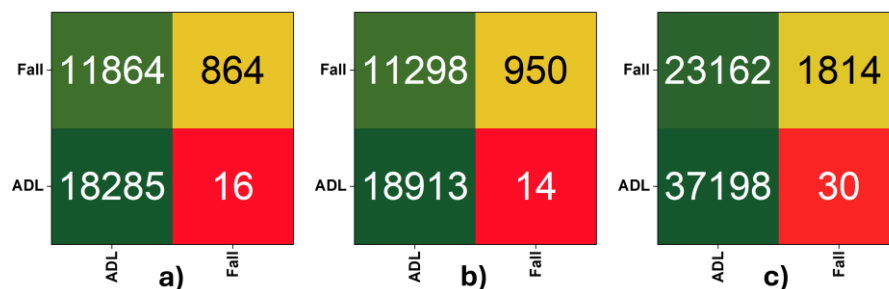


Figure 2. Confusion matrices of the Mahalanobis pre-filter:
(a) Group A; (b) Group B; (c) Combined result.

The Mahalanobis pre-filter was evaluated as the first decision stage of the proposed framework. In this stage, the positive class denotes a window forwarded to the CNN, while the negative class denotes a window filtered before CNN inference. Therefore, false-positive samples at this stage are not final false fall alarms; they are non-fall windows selected for further analysis by the second-stage classifier.

Figure 2 shows the confusion matrices of the Mahalanobis pre-filter. Figure 2(a) presents the result for Group A, Figure 2(b) presents the result for Group B, and Figure 2(c) presents the combined result of the two groups. In Group A, the pre-filter forwarded 864 fall windows to the CNN and missed 16. It also filtered 18,285 non-fall windows and forwarded 11,864 non-fall windows for second-stage classification. In Group B, the pre-filter forwarded 950 fall windows and missed 14. It filtered 18,913 non-fall windows and forwarded 11,298 non-fall windows. For the combined result, 1,814 fall windows were forwarded, 30 were missed, and 37,198 non-fall windows were filtered before CNN inference.

Table 2. Mahalanobis pre-filtering performance.

Evaluation set	Threshold	Fall recall	FPR	Specificity	Activation rate	Filtering rate
Group A	2.13	98.18%	39.35%	60.65%	41.02%	58.98%
Group B	2.07	98.55%	37.40%	62.60%	39.29%	60.71%
Combined	-	98.37%	38.37%	61.63%	40.15%	59.85%

Table 3 summarizes the main performance metrics derived from the confusion matrices. The selected thresholds were $\tau = 2.13$ for Group A and $\tau = 2.07$ for Group B. The fall recall values were 98.18% and 98.55%, respectively. These results show that most fall windows were retained for the CNN classifier. The combined fall recall was 98.37%, with a specificity of 61.63% and a filtering rate of 59.85%. The false-positive rates were 39.35% for Group A and 37.40% for Group B. These values indicate that some non-fall movements still passed the Mahalanobis stage. This

behavior is expected because some dynamic activities and transition windows exhibit statistical features similar to those of fall events. In the proposed two-stage structure, these windows are processed by the CNN rather than treated as final fall alarms.

4.2. Computational efficiency

The second-stage CNN was evaluated as the backend classifier for candidate windows selected by the Mahalanobis pre-filter. The CNN contains 26,946 trainable parameters and requires approximately 174K FLOPs per inference. In the validation experiment, the CNN achieved a precision of 96.59% and an F1-score of 95.98%. After quantization, the CNN was deployed on the ESP32-S3 MCU, where one inference required approximately 4.7 ms.

Table 3. Computational comparison between CNN-only inference and the proposed two-stage framework.

Metric	CNN-only baseline	Group A hybrid	Group B hybrid	Combined hybrid
Stage-1 MCU time	0	196 us	196 us	196 us
CNN time per activation	4.7 ms	4.7 ms	4.7 ms	4.7 ms
CNN activation rate	100%	41.02%	39.29%	40.15%
Average MCU time per window	4.7 ms	~2.12 ms	~2.04 ms	~2.08 ms

The Mahalanobis stage has a much lower computational cost than CNN inference. On the ESP32-S3, Stage 1 required 196 us per window, including min-max normalization, extraction of the nine statistical features, and Mahalanobis distance computation. Table 4 summarizes the measured execution times of the CNN-only baseline and the proposed hybrid framework. In the CNN-only baseline, every window is processed by the CNN, with an average time of 4.7 ms per window. In the hybrid framework, the Mahalanobis stage is executed for every window, while the CNN is activated only for candidate windows selected by the pre-filter. As shown in Table 4, the CNN activation rate decreased from 100% in the baseline to 41.02% in Group A and 39.29% in Group B. The corresponding average MCU time decreased to approximately 2.12 ms and 2.04 ms per window, respectively. In the combined setting, the CNN activation rate was 40.15%, and the average processing time per window was about 2.08 ms. These results show that the Mahalanobis pre-filter substantially reduces the number of CNN executions and lowers the average runtime on the MCU while preserving high first-stage recall.

4.3. Discussion

The experimental results indicate that the Mahalanobis distance serves as an effective lightweight pre-filter for embedded fall detection, reducing nine statistical features to a single distance value with a single decision threshold. This approach is simpler and more efficient than using multiple thresholds across axes and features, especially because it accounts for feature covariance arising from correlated human movement patterns. The threshold value allows for flexible deployment; lowering it can increase the number of windows sent to the CNN, enhancing safety for users at high risk of falls. Conversely, raising the threshold filters out more windows, potentially conserving energy for longer battery life. This means the same pre-filter structure can accommodate different user profiles without altering the feature extractor or CNN backend.

It is important to note the Mahalanobis pre-filter's role: it is not intended to make the final fall decision but to eliminate low-risk windows while retaining suspicious ones for CNN classification. As a result, false positives mainly increase the CNN's computational load, whereas false negatives are critical, as they can eliminate potential fall windows. This study has limitations, including the KFall dataset's focus on young, healthy subjects, whereas the application targets elderly care, where individuals may exhibit different gait patterns and fall dynamics. Future research should

test the framework with older adults, assess long-term energy consumption, and explore adaptive threshold selection for personalized deployment.

5. CONCLUSIONS

This paper presented a two-stage framework for real-time fall detection on wearable embedded devices. The proposed method uses Mahalanobis distance as a lightweight pre-filter to identify low-risk accelerometer windows before activating a compact CNN classifier. By combining nine statistical features into a covariance-aware distance score, the first stage provides a simple and adjustable screening mechanism for MCU-based deployment. The evaluation on the KFall dataset showed that the pre-filter preserved high fall sensitivity while substantially reducing the number of CNN executions. Hardware measurements on the ESP32-S3 further confirmed that the Mahalanobis stage incurs only a small runtime overhead compared with CNN inference, enabling the system to reduce average computation by about 60% relative to a CNN-only baseline. These results indicate that statistical pre-filtering can improve the efficiency of TinyML fall detection while still reserving neural classification for ambiguous movements. Future work will focus on validation with older adults, testing in daily-life conditions, long-term energy measurement, and adaptive threshold calibration for user-specific deployment.

Acknowledgement: Luong Cong Duan was funded by the PhD Scholarship Programme of Vingroup Innovation Foundation (VINIF), VinUniversity, code VINIF.2025.TS71. This research was funded by the Posts and Telecommunications Institute of Technology (PTIT), Hanoi, Vietnam, grant number 13-2026-HV-KTĐT1.

REFERENCES

- [1]. Z. Jiang et al., “Fall Detection Systems for Internet of Medical Things Based on Wearable Sensors: A Review”, *IEEE Internet Things J.*, vol. 11, no. 21, pp. 34797–34810, (2024). DOI: 10.1109/JIOT.2024.3421336.
- [2]. A. Ramachandran and A. Karuppiiah, “A Survey on Recent Advances in Wearable Fall Detection Systems”, *BioMed Research International*, vol. 2020, no. 1, p. 2167160, (2020). DOI: 10.1155/2020/2167160.
- [3]. J. Liu et al., “A review of wearable sensors based fall-related recognition systems”, *Engineering Applications of Artificial Intelligence*, vol. 121, p. 105993, (2023). DOI: 10.1016/j.engappai.2023.105993.
- [4]. C. V. F. Pereira, E. M. De Oliveira, and A. D. De Souza, “Machine Learning Applied to Edge Computing and Wearable Devices for Healthcare: Systematic Mapping of the Literature”, *Sensors*, vol. 24, no. 19, p. 6322, (2024). DOI: 10.3390/s24196322.
- [5]. H. Haresamudram, C. I. Tang, S. Suh, P. Lukowicz, and T. Plötz, “Past, Present, and Future of Sensor-based Human Activity Recognition Using Wearables: A Surveying Tutorial on a Still Challenging Task”, *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 9, no. 2, pp. 1–44, (2025). DOI: 10.1145/3729467.
- [6]. O. Aziz, M. Musngi, E. J. Park, G. Mori, and S. N. Robinovitch, “A comparison of accuracy of fall detection algorithms (threshold-based vs. machine learning) using waist-mounted tri-axial accelerometer signals from a comprehensive set of falls and non-fall trials”, *Med Biol Eng Comput*, vol. 55, no. 1, pp. 45–55, (2017). DOI: 10.1007/s11517-016-1504-y.
- [7]. J. He, Z. Zhang, X. Wang, and S. Yang, “A Low Power Fall Sensing Technology Based on FD-CNN”, *IEEE Sensors J.*, vol. 19, no. 13, pp. 5110–5118, (2019). DOI: 10.1109/JSEN.2019.2903482.
- [8]. X. Yu et al., “A practical wearable fall detection system based on tiny convolutional neural networks”, *Biomedical Signal Processing and Control*, vol. 86, p. 105325, (2023). DOI: 10.1016/j.bspc.2023.105325.
- [9]. J. Peng, X. Zhang, and H. Li, “A LSTM-Based Fall Prediction Method Using IMU”, *Advances in Asian Mechanism and Machine Science*, N. V. Khang, N. Q. Hoang, and M. Ceccarelli, Eds., *Mechanisms and Machine Science*, vol. 113, Cham: Springer International Publishing, pp. 680–690, (2022). DOI: 10.1007/978-3-030-91892-7_65.

- [10]. X. Hu, S. Yu, J. Zheng, Z. Fang, Z. Zhao, and X. Qu, “A hybrid CNN-LSTM model for involuntary fall detection using wrist-worn sensors”, *Advanced Engineering Informatics*, vol. 65, p. 103178, (2025). DOI: 10.1016/j.aei.2025.103178.
- [11]. T.-H. Nguyen et al., “A frame-level video annotation tool for dynamic gestures and apply in Vietnamese sign language”, *J. Mil. Sci. Technol.*, vol. 112, pp. 167–175, (2026). DOI: 10.54939/1859-1043.j.mst.112.2026.167-175.
- [12]. Y. Li et al., “A Decade of Progress in Wearable Sensors for Fall Detection (2015–2024): A Network-Based Visualization Review”, *Sensors*, vol. 25, no. 7, p. 2205, (2025). DOI: 10.3390/s25072205.
- [13]. S. R. Shakya, E. Ceh-Varela, J. Martinez, M. Fisher, and H. Allamezadeh, “Optimized CNN-Based Fall Detection for Elderly Using Wearable Devices on Resource-Constrained Devices”, 2025 IEEE 18th Dallas Circuits and Systems Conference (DCAS), Arlington, TX, USA: IEEE, pp. 1–6, (2025). DOI: 10.1109/DCAS65331.2025.11045488.
- [14]. V. Divya and R. L. Sri, “Docker-Based Intelligent Fall Detection Using Edge-Fog Cloud Infrastructure”, *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8133–8144, (2021). DOI: 10.1109/IJOT.2020.3042502.
- [15]. T. Sözer, “A cost-effective embeddable algorithm for accelerometer-based fall detector”, *Engineering Science and Technology, an International Journal*, (2025). DOI: 10.1016/j.jestch.2025.102185.
- [16]. N. B. Nguyen, Assoc. Prof. D. N. Pham Minh, A. Phan Huy, and Pham Hoang Hung, “Enhance micro-Doppler signatures-based human activity classification accuracy of FMCW radar using the threshold method”, *J. Mil. Sci. Technol.*, vol. 95, pp. 20–28, (2024). DOI: 10.54939/1859-1043.j.mst.95.2024.20-28.
- [17]. G. J. McLachlan, “Mahalanobis distance”, *Reson*, vol. 4, no. 6, pp. 20–26, (1999). DOI: 10.1007/BF02834632.
- [18]. X. Yu, J. Jang, and S. Xiong, “A Large-Scale Open Motion Dataset (KFall) and Benchmark Algorithms for Detecting Pre-impact Fall of the Elderly Using Wearable Inertial Sensors”, *Front. Aging Neurosci.*, vol. 13, p. 692865, (2021). DOI: 10.3389/fnagi.2021.692865.

TÓM TẮT

Giảm chi phí tính toán cho thiết bị phát hiện ngã sử dụng bộ lọc tiền xử lý Mahalanobis

Nghiên cứu này trình bày một khung hai giai đoạn cho việc phát hiện ngã trên các thiết bị nhúng đeo được. Phương pháp đề xuất sử dụng khoảng cách Mahalanobis làm bộ tiền lọc nhẹ để xác định các cửa sổ gia tốc rủi ro thấp trước khi kích hoạt bộ phân loại CNN. Bằng cách kết hợp 9 đặc trưng thống kê thành một giá trị khoảng cách có tính đến hiệp phương sai, giai đoạn đầu cung cấp một cơ chế sàng lọc đơn giản và có thể tùy chỉnh để triển khai trên MCU. Đánh giá trên bộ dữ liệu KFall cho thấy bộ tiền lọc vừa duy trì độ nhạy ngã cao, vừa giảm đáng kể số lần thực thi CNN. Các phép đo phân cứng trên ESP32-S3 xác nhận giai đoạn Mahalanobis chỉ tốn một lượng chi phí thời gian chạy rất nhỏ so với suy luận CNN, giúp hệ thống giảm khoảng 60% tính toán trung bình so với phương án chỉ dùng CNN. Kết quả này chỉ ra rằng tiền lọc thống kê có thể nâng cao hiệu suất phát hiện ngã của TinyML mà vẫn giữ lại phân loại nơ-ron cho các chuyển động mơ hồ. Nghiên cứu tương lai sẽ tập trung thử nghiệm trên người cao tuổi, đánh giá trong đời sống hàng ngày, đo lường năng lượng dài hạn và hiệu chuẩn ngưỡng thích ứng cho từng người dùng.

Từ khoá: Phát hiện ngã; Khoảng cách Mahalanobis; TinyML; Tính toán tại biên; Thiết bị đeo; Tính toán hiệu quả.