ITC 3/55
Information Technology and Control
Vol. 55 / No. 3/ 2026
pp. 1172-1189
DOI 10.5755/j01.itc.55.3.44384

A Cascaded CNN-ResNet Architecture for Computation-Aware Fall Detection on TinyML Wearable Devices

Received 2026/02/25 | Accepted after revision 2026/06/28

HOW TO CITE: Luong-Cong, D., Nguyen-Ngoc, M., Truong-Cao, D., Tran-T-Thuc, L. (2026) A Cascaded CNN-ResNet Architecture for Computation-Aware Fall Detection on TinyML Wearable Devices. *Information Technology and Control*, 55(3), 1172-1189. https://doi.org/10.5755/j01.itc.55.3.44384

A Cascaded CNN-ResNet Architecture for Computation-Aware Fall Detection on TinyML Wearable Devices

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

Faculty of Electronics Engineering I & EDA Lab, Posts and Telecommunications Institute of Technology, Hanoi, Vietnam; e-mails: duanlc@ptit.edu.vn, minhnn@ptit.edu.vn, dungtc@ptit.edu.vn, linhtt@ptit.edu.vn

Corresponding author: duanlc@ptit.edu.vn

Falls among older adults are a major public health concern, yet accurate fall detection on resource-constrained wearable devices remains challenging because high-performance deep models often require continuous computation. This study proposes a computation-aware cascaded CNN-ResNet architecture for TinyML-based wearable fall detection. A lightweight CNN processes four accelerometer-derived channels, namely x, y, z, and root-mean-square (RMS), to perform initial classification and extract 32-dimensional embeddings. These embeddings are stored in a first-in, first-out (FIFO) buffer. When the CNN confidence for Activities of Daily Living (ADL) classification falls below a threshold of 0.995, a compact ResNet is activated to analyze six consecutive embeddings for refined spatiotemporal classification. The system was evaluated on the KFall dataset using leave-one-subject-out (LOSO) cross-validation across 32 subjects. It achieved 98.97% ± 0.39% accuracy, 98.48% ± 0.91% precision, 97.96% ± 0.95% recall, and 98.21% ± 0.62% F1-score. On-device benchmarking on ESP32-S3 and STM32F407 MCUs confirmed a combined quantized model size of 0.46 MB and average latencies of 12.3–17.3 ms and 9.6–14.1 ms, respectively, under a conservative daily-activity scenario with 3–5% ResNet invocation. Even under full ResNet activation, the maximum latency remained below 252 ms, supporting real-time on-device inference within the 500 ms sliding-window interval. These results indicate that the proposed cascade provides an effective trade-off between detection accuracy and average computational cost for MCU-based wearable fall detection.

KEYWORDS: fall detection, wearable devices, computation-aware, cascaded inference, TinyML, real-time.

1. Introduction

Falls are a major public health concern, particularly as the global population aged 60 years and over is projected to reach 2.1 billion by 2050 [32], [44]. They are the second leading cause of unintentional injury-related deaths worldwide and cause approximately 684,000 fatalities each year [45]. Beyond physical injuries, such as fractures, traumatic brain injuries, and reduced mobility, falls also lead to serious secondary risks [42]. Previous studies show that many older adults who remain on the floor for a long period after a fall face a high risk of death within the following months [5], [41]. Falls also cause psychological problems, including anxiety, depression, and fear of falling, which may reduce independence and accelerate functional decline [8]. These physical, psychological, and economic impacts underscore the need for accurate, timely fall detection systems (FDS) [22].

Fall detection technologies can be divided into ambient monitoring systems and wearable sensor systems [14]. Ambient systems are effective in controlled indoor environments, but they may raise privacy concerns and have limited coverage [27], [28]. Wearable fall detection systems (WFDSs) are more suitable for continuous monitoring in daily life because they are portable and can operate close to the user [5], [22]. These systems commonly use inertial measurement units (IMUs), including accelerometers and gyroscopes, to capture motion patterns associated with falls. Existing algorithms have evolved from threshold-based methods (TBM) to classical machine learning (ML) models and deep learning (DL) models [27]. DL models, such as CNNs, LSTMs, and GRUs, can learn discriminative features from raw sensor data, thereby improving detection accuracy. However, they usually require high memory and processing resources, which limits their deployment on battery-powered microcontrollers (MCUs). Cloud and edge processing can reduce the local computing burden, but they may introduce latency and privacy risks [37], [43]. TinyML provides a practical direction by enabling compressed models on MCUs through techniques such as quantization and pruning [1], [47]. However, a clear trade-off still exists between computational efficiency and detection performance. Lightweight models reduce computational cost, but they may increase false alarm rates. In contrast, larger models can improve accuracy but often entail higher computational costs.

Several adaptive inference paradigms have been proposed to reduce computational cost in resource-constrained edge and wearable systems. Early exit networks, such as BranchyNet, use auxiliary classifiers at intermediate layers and allow simple samples to exit early based on confidence thresholds [34], [40]. However, these models do not explicitly exploit temporal context and are not commonly validated on MCU-class hardware. Traditional cascaded classifiers process inputs through independent models, but they usually do not preserve intermediate representations across windows [39]. As a result, they cannot fully use temporal information from consecutive sensor windows. Recent TinyML approaches, such as TinyFallNet [25] and PreFallKD [10], improve model compactness through knowledge distillation and quantization [47], [25], [23]. Nevertheless, they often use a fixed inference path, which produces a constant computational load regardless of input complexity. Other hybrid systems use a lightweight trigger followed by a deeper classifier, but many of them depend on cloud processing [43]. This may introduce latency and privacy concerns that are not suitable for time-critical fall detection.

To address these limitations, this study proposes a computation-aware cascaded CNN-ResNet architecture that combines a lightweight CNN gatekeeper with a compact ResNet refinement stage. The CNN extracts 32-dimensional embeddings from each accelerometer window and pushes them into a FIFO buffer. When the CNN confidence for routine ADL recognition falls below a predefined threshold, the ResNet analyzes recent embeddings as a compact spatiotemporal representation. This selective feature-reuse design reduces average computation while preserving fall detection performance. The main contributions of this study are summarized as follows:

- A computation-aware cascaded CNN-ResNet architecture is proposed for wearable fall detection. It combines a lightweight CNN gatekeeper with a compact ResNet expert and uses a FIFO buffer to transfer six consecutive 32-dimensional CNN embeddings for context-aware classification.

- A confidence-based activation mechanism is proposed to balance fall recall, ADL precision, and computational cost. The selected threshold is characterized using the ADL-side trigger rate and fall miss rate, enabling selective ResNet activation and reducing average latency to 9.6–17.3 ms per 500 ms sliding-window interval under a conservative ADL scenario.
- The system is evaluated using LOSO cross-validation on 32 subjects from the KFall dataset. It achieves 98.97% ± 0.39% accuracy, 98.48% ± 0.91% precision, 97.96% ± 0.95% recall, and 98.21% ± 0.62% F1-score. A paired t-test confirms a significant improvement over the standalone CNN baseline.
- The quantized cascaded model, measuring 0.46 MB, is deployed on ESP32-S3 and STM32F407 MCUs. The maximum latency remains below 252 ms even when ResNet is fully invoked, supporting real-time on-device inference within the 500 ms sliding-window interval.

The remainder of this paper is organized as follows. Section 2 situates the proposed approach within existing fall detection and TinyML literature. Section 3 details the cascaded CNN-ResNet architecture and the confidence-based trigger mechanism. Sections 4 and 5 describe the experimental protocol and report LOSO results alongside on-device benchmarking. Section 6 interprets the findings and acknowledges limitations, and Section 7 concludes the paper.

2. Related Work

Wearable fall detection has been studied through several algorithmic directions, ranging from threshold-based methods to DL models [27]. Early systems mainly used thresholds on acceleration magnitude, angular velocity, or impact-related features to detect abrupt body motion [4], [36], [38]. These methods are attractive for embedded devices because they require little memory and computation. However, their performance is sensitive to sensor placement, signal noise, user habits, and fixed parameter settings. Classical ML methods, such as support vector machines, k-nearest neighbors, decision trees, random forests, and XGBoost, improved robustness by using handcrafted statistical and frequency-domain features from inertial signals [2], [6], [7]. Nevertheless, these approaches still depend strongly on feature design and may not capture the temporal patterns that separate falls from fast daily activities [31].

DL has reduced the need for manual feature engineering by learning representations directly from sensor data [14]. CNN models are effective for local temporal patterns in acceleration signals, while LSTM and GRU models capture longer sequential dependencies [3], [13], [15]. Recent end-to-end models have achieved high accuracy on public fall detection datasets [12]. However, these models usually require higher memory, more computation, and continuous inference at full capacity. This limits their practical use on battery-powered MCUs, especially when a wearable device must operate continuously under strict latency and power constraints [9], [35].

TinyML provides an important direction for bringing ML/DL to MCU-based healthcare devices. Quantization, pruning, and knowledge distillation can reduce model size and inference cost while preserving useful accuracy [10], [25], [47]. Beyond fall detection, recent healthcare IoT studies further demonstrate the importance of local and privacy-aware intelligence on constrained devices. For example, smartwatch-based blood glucose monitoring and IoT-enabled wheelchair systems have explored transfer learning and distributed learning to reduce dependence on centralized cloud processing [29], [30], [46]. Although these studies target different healthcare applications, they support the broader motivation for low-memory, privacy-aware, and autonomous inference in wearable or assistive systems. In fall detection, recent TinyML-oriented studies such as TinyFallNet [25], PreFallKD [10], and MicroFallNet [20] have demonstrated the feasibility of compact models for wearable or MCU-constrained deployment. However, these approaches generally follow a fixed inference path, in which every input window is processed with the same computational budget regardless of motion complexity.

Adaptive inference has been proposed to reduce average computation by allocating more processing only to difficult samples. Early exit networks,

such as BranchyNet and its variants, attach auxiliary classifiers to intermediate layers and allow high confidence samples to exit early [19], [34], [40]. This strategy is effective for some vision and edge computing tasks, but it typically operates within a single backbone and processes each input independently. It also does not preserve temporal representations across consecutive sensor windows. Cascaded classifiers follow another strategy, where a lightweight first model filters easy samples and a stronger second model handles uncertain cases [39]. This idea is suitable for wearable fall detection because most daily activity windows are simple. However, standard cascades often discard intermediate features after each decision. Some hybrid fall detection systems use a local trigger followed by a cloud-based classifier [43], [49]. These systems reduce local computation, but they may introduce communication latency, energy cost, and privacy risks.

Overall, prior studies have improved fall detection from different directions. Threshold and classical methods are lightweight but less robust. DL methods improve accuracy but increase computation. TinyML methods reduce memory cost but often keep a fixed inference path. Adaptive and cascaded inference can reduce average computation, but existing designs have limited use of cross-window feature reuse and limited validation on MCU-class wearable hardware. This study addresses this gap by combining confidence-based selective activation, FIFO-based reuse of CNN embeddings, and full on-device evaluation on ESP32-S3 and STM32F407 MCUs.

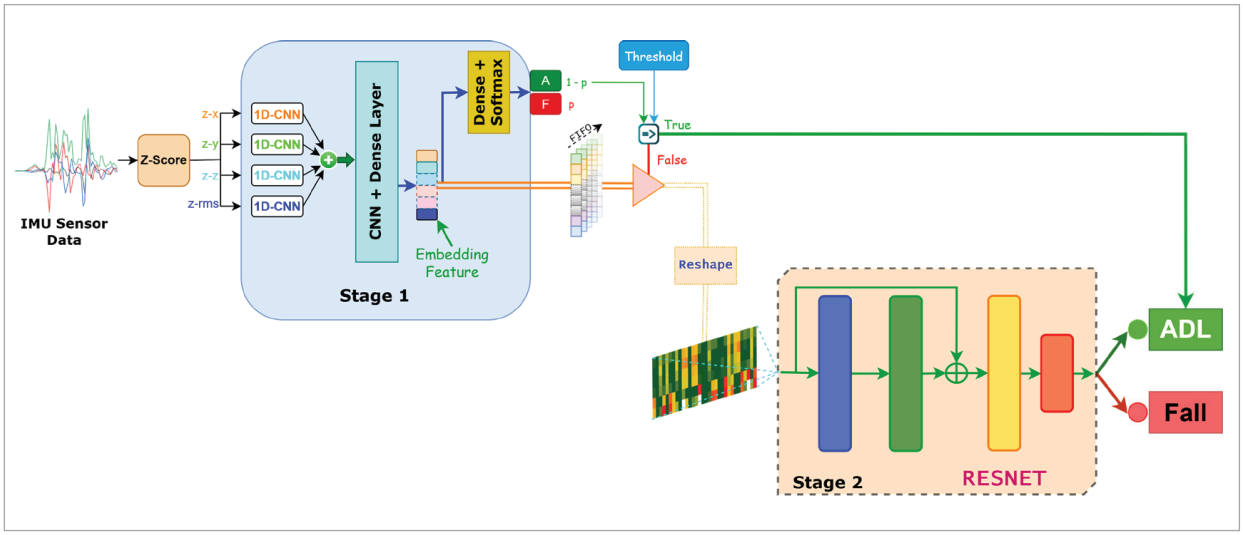
3. Proposed CNN-ResNet Architecture

3.1. System Overview

The proposed fall detection system uses a two-stage cascaded architecture designed for computation-aware inference, as shown in Figure 1. Stage 1 is a lightweight CNN that processes each incoming accelerometer window. It performs an initial ADL/fall classification and extracts a compact 32-dimensional embedding vector. Each new embedding is pushed into a FIFO buffer, which retains the most recent embeddings for Stage 2. The conditional trigger then checks the ADL confidence produced by Stage 1. If the ADL confidence is greater than or equal to θ, the system accepts the Stage 1 decision and does not activate the second stage. If the ADL confidence is below the selected threshold, Stage 2 is activated. The ResNet then receives the six most recent CNN embeddings from the FIFO buffer and produces the final ADL or fall decision.

Figure 1

Overview of the computation-aware cascaded CNN-ResNet architecture.



Algorithm 1: Computation-Aware Cascaded CNN-ResNet Inference

Input: x_t, θ
Output: y_t
$[e_t, p_t(ADL), p_t(Fall)] \leftarrow S1_CNN(x_t)$
$B_t \leftarrow [e_{t-5}, e_{t-4}, e_{t-3}, e_{t-2}, e_{t-1}, e_t]$
if $p_t(ADL) \geq \theta$ **then**
    $y_t \leftarrow ADL$
else
    $Z_t \leftarrow Reshape(B_t)\ to\ (6, 32, 1)$
    $[q_t(ADL), q_t(Fall)] \leftarrow S2_ResNet(Z_t)$
    $y_t \leftarrow argmax(q_t)$
end if
return y_t

This design separates simple and uncertain cases. Clear ADL windows are processed only by the lightweight CNN, while the stronger ResNet classifier analyzes windows with lower ADL confidence. Therefore, the system can reduce the average computational cost while preserving temporal context for difficult motion patterns. The overall inference flow is summarized in Algorithm 1.

The following subsections detail the CNN stage and feature extraction (Section 3.2), the conditional trigger mechanism (Section 3.3), and the ResNet-based classification stage (Section 3.4).

3.2. Stage 1: Lightweight CNN

Stage 1 receives windowed accelero23meter data and produces both an initial class prediction and an embedding vector for Stage 2. The input contains four channels derived from the triaxial accelerometer signals: the x, y, and z axes, and the RMS magnitude. The RMS value at time step t is defined as:

$$RMS = \sqrt{a_x^2 + a_y^2 + a_z^2}. \quad (1)$$

Gyroscope channels are not used in this study to reduce sensor power consumption while retaining useful fall detection information from acceleration signals. Each four-channel input window is normalized by z-score normalization before being passed to the CNN:

$$x_i' = \frac{x_i - \mu}{\sigma}, \quad (2)$$

where μ and σ denote the mean and standard deviation computed over the training data of each channel. In each LOSO fold, these values are computed only from the training subjects to avoid information leakage from the held-out test subject.

The Stage 1 CNN first applies four parallel 1D convolution branches, one for each input channel. Each branch uses 16 filters, a kernel size of 33 samples, and a stride of 7 samples. At 50 Hz, a 33-sample kernel corresponds to about 0.66 s, which is long enough to capture short acceleration patterns associated with abrupt posture changes, as supported by prior fall detection studies [24], [33]. The branch outputs are concatenated and reshaped for 2D convolution. The following 2D convolution uses 32 filters with a kernel size of (4, 3), matching the output shape and parameter count reported in Table 1. The resulting features are passed through dense layers to obtain the embedding and final softmax output. The complete Stage 1 architecture is summarized in Table 1.

Stage 1 produces two outputs. The first output is the 32-dimensional embedding vector extracted from the penultimate dense layer:

$$e_t = f_{emb}(x_t) \in R^{32}. \quad (3)$$

Table 1
Lightweight CNN Model Architecture.

Layer	Output	Param #
Input	(N, 100, 4)	0
Channel Convolutions (x4)	(N, 10, 1, 16)	2,176
Concat & Reshape	(N, 4, 10, 16)	0
Conv2D	(N, 1, 8, 32)	6,176
Flatten & Dense	(N, 64)	16,448
Dense (*emb*)	(N, 32)	2,080
Output (Dense)	(N, 2)	66
Total Parameters:		**26,946**

The second output is the softmax probability vector for ADL and fall classes:

$$\left[p_t(ADL), p_t(Fall)\right] = softmax\left(f_{cls}(e_t)\right). \quad (4)$$

The lightweight CNN contains 26,946 parameters. This compact size makes it suitable for continuous first-stage screening on MCU-based wearable devices.

3.3. Conditional Trigger Mechanism

The conditional trigger controls whether the system accepts the Stage 1 output or activates the Stage 2 ResNet. At each time step t, the embedding e_t from Stage 1 is pushed into the FIFO buffer. The buffer stores the six most recent embeddings and forms the temporal context matrix:

$$B_t = \left[e_{t-5}, \ldots, e_t\right] \in R^{6\times 32}. \quad (5)$$

The trigger uses the ADL confidence $p_t(ADL)$ and a predefined threshold (θ). The final decision is defined as:

$$\hat{y}_t = \begin{cases} f_{ResNet}(B_t), & if\ p_t(ADL) < \theta \\ ADL, & if\ p_t(ADL) \geq \theta \end{cases}. \quad (6)$$

If $p_t(ADL)$ is greater than or equal to θ, the system treats the window as a confident ADL and skips Stage 2. If $p_t(ADL)$ is lower than θ, the system sends the FIFO matrix B_t to Stage 2. The ResNet then produces the final ADL/fall prediction, serving as a refinement classifier for low-confidence windows rather than a fall-only confirmation module.

In this study, θ is set to 0.995. This value was selected via a nested LOSO validation on training folds as detailed in the experimental evaluation. The trigger primarily routes inputs with low ADL confidence to the ResNet, thereby allocating additional computation to uncertain or fall-related motion patterns rather than to all inputs.

3.4. Stage 2: ResNet Classifier

When triggered, Stage 2 uses a compact ResNet [16] to refine the classification result. The input to the ResNet is the FIFO matrix B_t reshaped as a tensor of size (6, 32, 1). The six rows represent six consecutive CNN embeddings and cover up to 4.5 s of motion history. The 32 columns represent the learned embedding components from Stage 1. Although this tensor is not a natural image, it can be treated as a compact two-dimensional feature map. The residual convolutions learn local relations in both the temporal and embedding dimensions. This allows Stage 2 to model the progression of a fall-related motion rather than relying on a single window.

Table 2
ResNet Architecture Specification.

Layer Block	Output	Param #
Input	(N, 6, 32, 1)	0
Initial Conv2D + BatchNorm	(N, 6, 32, 32)	416
Residual Block 1, 32 filters (x2)	(N, 6, 32, 32)	37,376
Residual Block 2, 64 filters (x2)	(N, 3, 16, 64)	132,352
Residual Block 3, 128 filters (x1)	(N, 2, 8, 128)	230,912
Global Average Pooling + Dropout	(N, 128)	0
Dense + BatchNorm	(N, 32)	4,256
Output (Dense + Softmax)	(N, 2)	66
Total Parameters:		**405,378**

The ResNet architecture includes an initial convolution layer, residual blocks with 32, 64, and 128 filters, global average pooling, dropout, a dense layer, and a final softmax classifier. Batch normalization is used in the convolutional and dense blocks to stabilize training. The full architecture is summarized in Table 2.

With 405,378 parameters, this second-stage classifier is larger than the Stage 1 CNN; however, it is invoked only when the Stage 1 ADL confidence falls below the selected threshold. This selective activation keeps the average inference cost low during continuous monitoring. The choice of ResNet as the Stage 2 backbone is further evaluated through the ablation study in Section 5.4.

4. Experimental Setup

4.1. Dataset and Preprocessing

This study used the publicly available KFall dataset [48]. The dataset contains inertial data from 32 young adult participants performing 21 ADLs and 15 simulated falls. The falls were simulated by healthy young adults, not real falls from older users. Therefore, the results should be interpreted with caution when considering clinical elderly populations. The inertial sensor was placed at the lower back, which is a common position for fall detection. Although KFall provides accelerometer, gyroscope, and orientation signals, this study used only four accelerometer-based channels: x, y, z, and RMS magnitude. This setting reduces sensing cost while preserving motion information useful for fall detection.

The raw signals were recorded at 100 Hz and processed in four steps. First, the signals were downsampled to 50 Hz to reduce computation while preserving the main frequency content of fall movements. Second, the continuous signals were segmented into 2-s windows with a 0.5-s sliding step [33]. Third, each four-channel window was normalized using z-score normalization. Fourth, each window was assigned a binary label using the fall phase annotation in KFall. A window was labeled as fall if at least 50% of its samples were annotated as the falling phase. If the falling phase occupied less than 50% of the window, the window was labeled as ADL [24].

After preprocessing, the dataset contained 51,639 ADL windows and 10,565 fall windows. This corresponds to an imbalanced distribution in which ADL windows are about five times more frequent than fall windows. Therefore, accuracy alone may not fully reflect model quality, and precision, recall, and F1-score are emphasized in the evaluation.

4.2. Evaluation Protocol

The proposed system was evaluated using LOSO cross-validation. In each of the 32 folds, data from 31 subjects were used for training, and data from the remaining subject were used for testing. All reported classification metrics were computed at the window level, with fall treated as the positive class. The final results were reported as the mean and standard deviation across all LOSO folds.

Four standard metrics were used to evaluate classification performance. Accuracy measures the overall ratio of correctly classified windows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{7}$$

Recall measures the ratio of true fall windows that are correctly detected:

$$Recall = \frac{TP}{TP + FN}. \tag{8}$$

Precision measures the ratio of predicted fall windows that are true fall windows:

$$Precision = \frac{TP}{TP + FP}. \tag{9}$$

The F1-score is the harmonic mean of precision and recall:

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{10}$$

Because the dataset is imbalanced, precision, recall, and F1-score are more informative than accuracy alone. Recall is important because missed falls are safety critical, while precision is important because false alarms can reduce user trust. To compare the proposed cascaded model with the standalone CNN baseline, a paired t-test was applied to the F1-scores obtained from the 32 LOSO folds. Cohen's *d* was also reported as the effect size.

Two additional metrics were used to analyze the cascade's computation-aware behavior. The trigger rate was expressed on the ADL side as:

$$Trigger\ Rate = \frac{FP}{TN + FP}. \tag{11}$$

The fall miss rate was expressed as:

$$Miss\ Rate = \frac{FN}{TP + FN}. \tag{12}$$

Here, (TN + FP) is the total number of ADL windows, and (TP + FN) is the total number of fall windows. These metrics quantify the trade-off between ADL false-trigger behavior and missed fall windows.

4.3. Ablation Study Design

Two ablation studies were conducted to validate the main design choices of the proposed architecture. The first ablation examined the effect of FIFO buffer length. The number of consecutive embeddings stored in the buffer was varied across five settings: 4, 5, 6, 7, and 8. All other preprocessing and model settings were kept unchanged. This experiment was used to determine the amount of temporal context required by Stage 2 for reliable, refined classification.

The second ablation evaluated the Stage 2 backbone. The Stage 1 CNN was kept fixed, and the same FIFO embedding sequences were used for all candidate backbones. Only the Stage 2 backbone was replaced. Three lightweight models, including LeNet-5 [26], MobileNetV1 [18], and SqueezeNet [21], were compared with ResNet under the same LOSO protocol. Each candidate was evaluated using classification performance, parameter count, quantized model size, and on-device inference latency. For the backbone ablation, latency was measured on the ESP32-S3 MCU. This setting supports a fair comparison of Stage 2 candidates under the same embedded deployment target.

4.4. MCU Deployment Setup

To evaluate practical deployment feasibility, the trained models were quantized and deployed on two representative MCUs: the ESP32-S3 with an Xtensa LX7 dual-core processor at 240 MHz and the STM32F407 with an ARM Cortex-M4 processor at 168 MHz. These platforms represent two common embedded targets for TinyML systems. The trained models were converted to TensorFlow Lite for Microcontrollers [11] format and then quantized to 8-bit integer precision. The quantized models were compiled into C source code arrays for execution on the devices.

Platform-specific acceleration libraries were used to improve inference speed. ESP-NN was used for the ESP32-S3 with the ESP-IDF toolchain, while CMSIS-NN was used for the STM32F407 with ARM Compiler v6. All firmware builds were compiled with the -O2 optimization level. The deployment evaluation measured flash usage, peak RAM consumption, and inference latency. Latency was averaged over 10,000 runs for both Stage 1 and Stage 2.

System-level average latency was evaluated under two operating scenarios. The first scenario represents daily monitoring, where most windows are ADL windows and only a small portion activates the ResNet. Based on the trigger rate analysis across activity groups in Section 5.3, routine ADLs typically yield very low ResNet invocation rates. In contrast, abrupt ADL patterns and fall-adjacent segments yield higher rates. Therefore, a 3–5% ResNet invocation range was selected as a conservative ADL operating scenario. The second scenario represents the worst-case fall condition, where the ResNet is invoked for all windows. These two scenarios show both the expected average cost during daily use and the upper latency bound when the second stage is fully activated.

5. Results and Analysis

5.1. Overall LOSO Performance

The proposed cascaded CNN-ResNet system achieved strong, stable performance under the LOSO cross-validation protocol. Across 32 folds, the model reached an accuracy of 98.97% ± 0.39%, precision of 98.48% ± 0.91%, recall of 97.96% ± 0.95%, and F1-score of 98.21% ± 0.62%. The 95% confidence interval (95% CI) of the F1-score was 97.99% to 98.43%, showing that the system maintained a consistent balance between missed fall reduction and false alarm control across unseen subjects. As shown in Figure 2, the metric distributions are tightly concentrated across LOSO folds, indicating that the proposed cascade provides stable subject-independent performance rather than isolated improvements in only a few subjects.

The aggregated confusion matrix in Figure 3 also supports the overall result. The cascaded model correctly detected 98.0% of fall windows and reduced the missed fall rate to about 2.0%. At the same time, ADL recognition remained high, with about 99.4% of ADL windows assigned to the ADL class. These results indicate that the system improves fall sensitivity while keeping false alarms low.

Figure 2
Comparative performance metrics of standalone CNN and proposed CNN-ResNet.

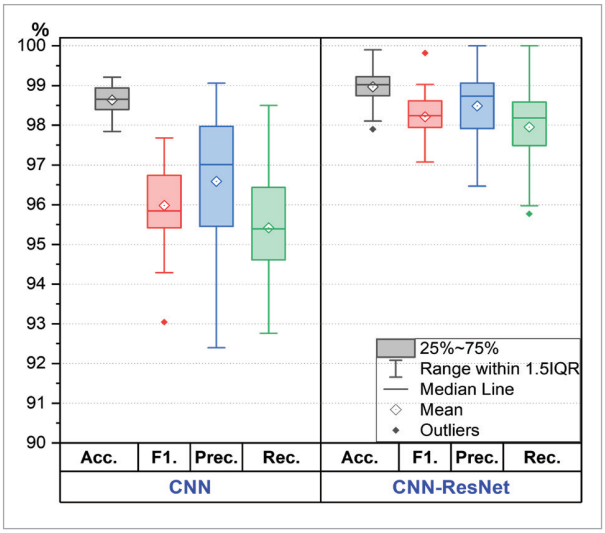
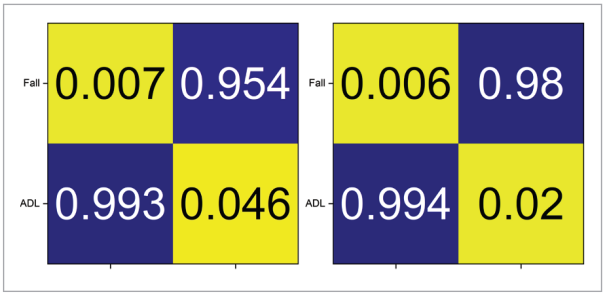


Figure 3
Aggregated confusion matrices (CNN and CNN-ResNet).



The subject-wise results in Figure 4 further show that the F1-score remained above 97% for all subjects, with the lowest values observed for P7, P23, and P24. These cases showed different error tendencies rather than a single failure mode. P7 exhibited a relatively balanced reduction in precision and recall, suggesting comparable contributions from false alarms and missed falls. P23 showed lower precision while maintaining recall, suggesting that some subject-specific ADL patterns were more likely to be misclassified as falls. Conversely, P24 maintained high precision but showed lower recall, suggesting that some fall windows were less distinguishable from ADL windows and were therefore missed. These patterns highlight the difficulty of subject-independent LOSO evaluation. They are consistent with the trigger analysis, which found that abrupt ADLs, such as picking up an object, collapsing into a chair, and lying down quickly, produced higher ResNet activation rates because of their similarity to fall-related motions. At the upper end, several subjects achieved F1-scores above 99%, indicating reliable classification for well-separated motion patterns.

5.2. Cascaded Architecture Effectiveness

This experiment examines whether the second-stage ResNet provides a statistically meaningful improvement over the lightweight CNN gatekeeper alone. The proposed CNN-ResNet model was compared with the standalone CNN baseline under the same LOSO setting. As shown in Figure 2, the cascaded model improved all four metrics. The F1-score increased from 95.98% to 98.21%, while precision increased from 96.59% to 98.48%. These results confirm that the ResNet refinement stage enhances overall fall detection performance while reducing false alarms.

The confusion matrices in Figure 3 show the same trend. The standalone CNN detected 95.4% of fall windows and missed 4.6%, while the cascaded CNN-ResNet detected 98.0% of fall windows and missed 2.0%. This improvement is important because missed falls are the most critical error type in fall monitoring. The paired t-test on the F1-scores across the 32 LOSO folds confirmed that the improvement was statistically significant, with $t(31) = 12.038$, $p < 0.0001$, and Cohen's $d = 2.128$. Therefore, the performance gain is consistent across subjects and is not driven by only a small number of favorable cases.

5.3. Trigger Mechanism Analysis

This analysis examines whether the confidence threshold can reduce missed fall windows while keeping ResNet invocation low for routine ADLs. The confidence threshold that controls Stage 2 activation was selected using a nested LOSO procedure. Candidate threshold values from $\theta = 0.90$ to $\theta = 1.00$ were evaluated within the training portion of each LOSO fold to avoid test-subject leakage. Four indicators were analyzed: ADL precision, fall recall, ResNet trigger rate, and fall miss rate, as shown in Figure 5.

ADL precision remained high over the full threshold range, changing only slightly from about 99.4% to 99.7%. In contrast, fall recall increased as θ in-

Figure 4

Per-subject performance metrics in LOSO validation using CNN-ResNet.

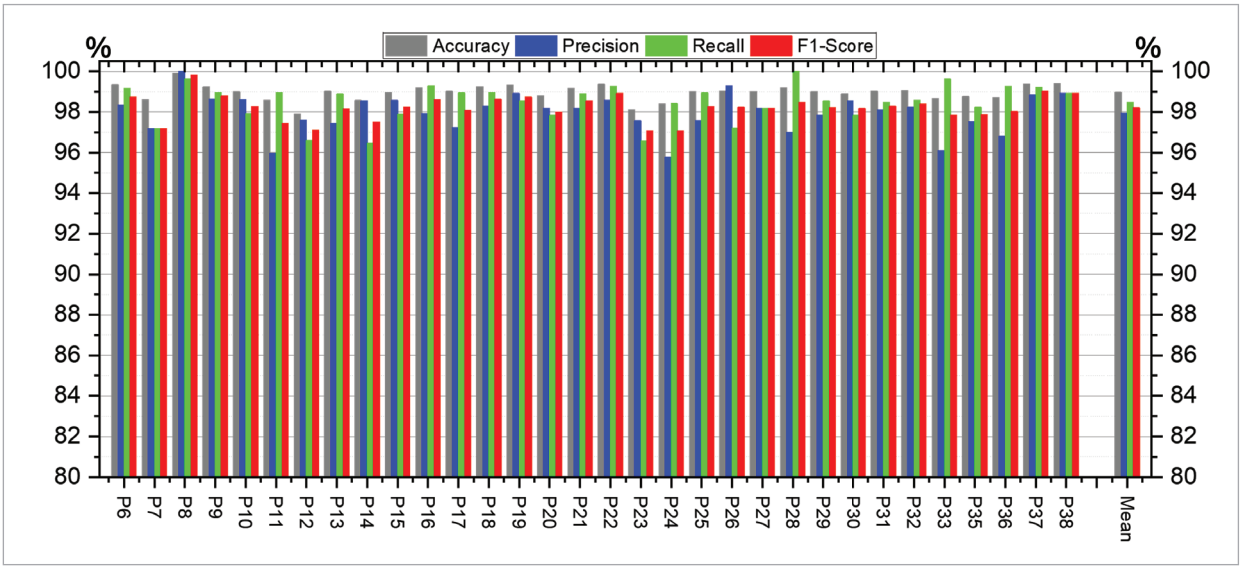
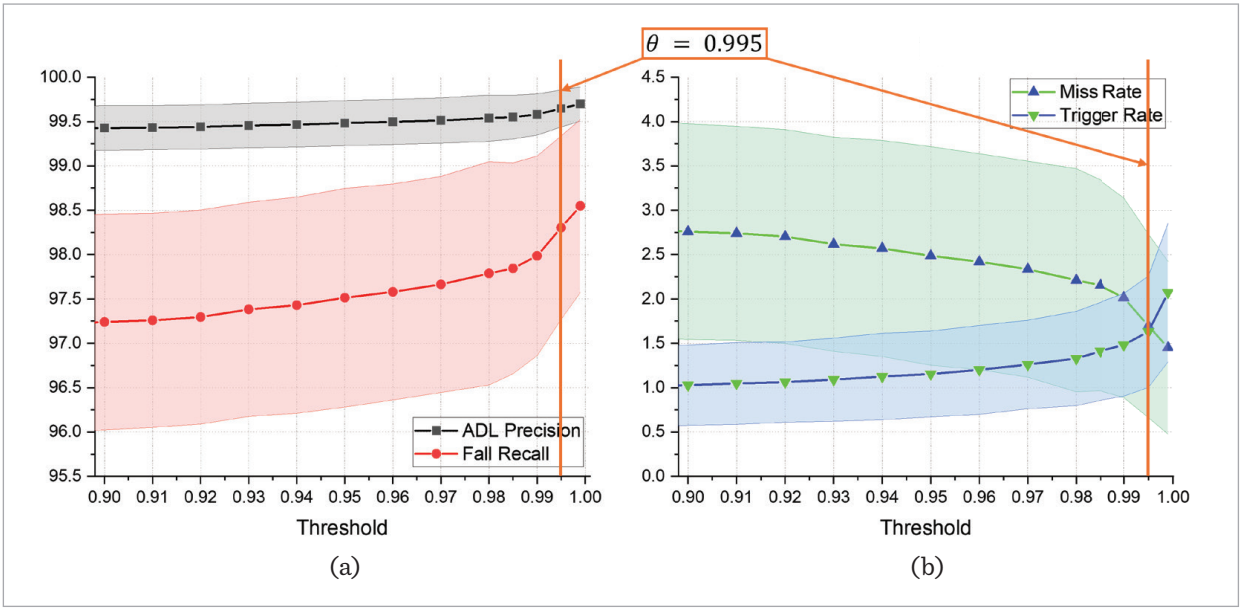


Figure 5

Threshold-risk analysis of the confidence-based trigger mechanism: (a) ADL precision and fall recall; (b) fall miss rate and ResNet trigger rate.



creased. It rose from about 97.2% at θ = 0.90 to about 98.5% at θ = 0.995. This trend is expected because a higher threshold sends more low-confidence windows to Stage 2, which reduces missed fall cases. The trigger rate also increased gradually from about 1.0% at θ = 0.90 to about 1.6% at θ = 0.995, while the miss rate decreased from about 2.75% to about 1.5%.

The threshold θ = 0.995 was selected as the final operating point because it provided a good balance between fall recall and computation. Increasing the threshold to θ = 1.00 produced only a small addition-

Figure 6
Trigger rates for various daily and fall-adjacent activities.

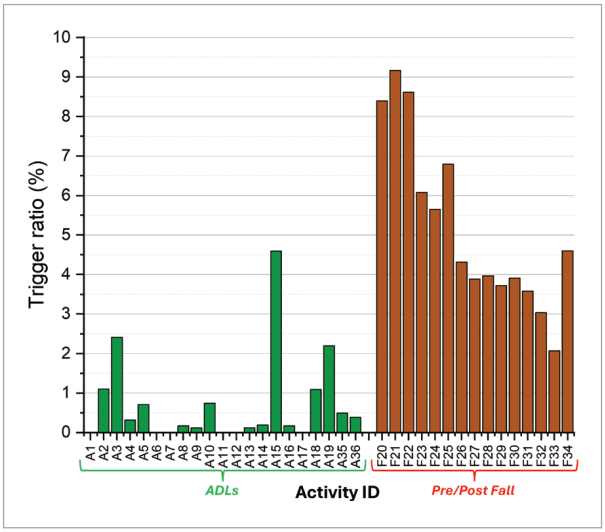
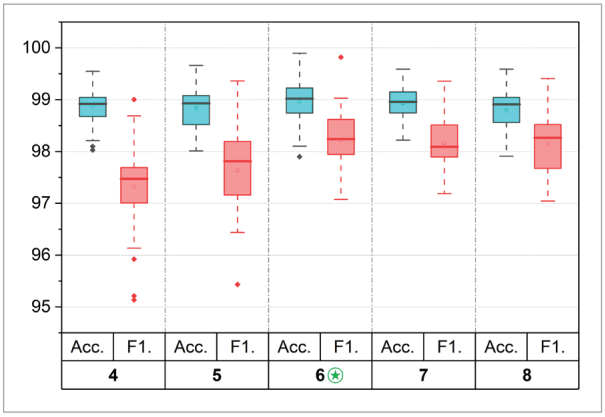


Figure 7
Classification performance across FIFO buffer lengths.



al reduction in miss rate, but it also increased the variability of the trigger behavior across subjects. Therefore, θ = 0.995 was selected as a stable and practical operating point rather than the most aggressive setting.

The per-activity trigger analysis in Figure 6 further shows that Stage 2 activation follows the motion complexity of each activity. Most routine ADLs had trigger rates below 1%, indicating that the CNN classified simple daily movements with high confidence. The highest ADL trigger rate was observed for A15 (collapse into a chair), at about 4.6%, followed by A3 (pick up an object from the floor), at about 2.2%, and A19 (lie down on the bed quickly), at about 1.1%. These activities include abrupt posture changes and are closer to fall patterns than normal walking or standing activities. For the fall-adjacent segments F20 to F34, trigger rates ranged from about 3% to more than 9%, confirming that the trigger sends more complex, risk-related motions to the ResNet.

Based on this analysis, the 3%–5% invocation range used in the MCU benchmark was chosen as a conservative ADL operating scenario. This range is higher than the trigger rate of most routine ADLs and is closer to more difficult activities. It therefore gives a slightly stricter latency estimate than a purely average daily activity setting, while still reflecting realistic selective activation behavior.

5.4. Ablation Studies

The ablation studies test two design choices that directly affect temporal context and embedded cost: FIFO length and Stage 2 backbone. The first ablation study evaluated the FIFO buffer length. The number of consecutive embeddings was varied from 4 to 8, as shown in Figure 7. Buffer lengths of 4 and 5 produced lower F1-scores and wider variability, suggesting that the ResNet received insufficient temporal context. A buffer length of 6 achieved the highest mean F1-score of 98.21% ± 0.62% and the most stable distribution. Six embeddings cover up to 4.5 s of raw motion history, providing sufficient temporal information to distinguish falls from fast ADLs. Increasing the buffer length to 7 or 8 did not improve performance and slightly reduced consistency, possibly because older embeddings added less relevant information. Therefore, a FIFO length of 6 was used in the final system.

To assess the impact of the Stage 2 backbone, LeNet-5, MobileNetV1, SqueezeNet, and ResNet were compared using the same FIFO embedding input and LOSO protocol. The results are shown in Table 3. ResNet achieved the highest accuracy and F1-score, while SqueezeNet achieved a very close F1-score with lower latency. This indicates a clear trade-off between maximum accuracy and lower Stage 2 latency.

Although ResNet has the highest Stage 2 inference latency among the tested backbones, this cost is acceptable in the proposed cascade because Stage 2 is activated only for a small portion of windows. The latency is therefore amortized over many low-cost

Table 3
Performance of candidate Stage 2 backbones.

Model	Acc. (%)	F1. (%)	Param #	Size (KB)	Inf. (ms)
LeNet-5	98.33 ± 0.55	97.10 ± 0.89	61,026	70.6	43
MobileNetV1	98.80 ± 0.39	97.90 ± 0.71	281,874	364.2	39
SqueezeNet	98.91 ± 0.40	98.11 ± 0.68	407,186	500.3	192
ResNet	**98.97 ± 0.39**	**98.21 ± 0.62**	**405,378**	**435.5**	**247**

Table 4
Embedded inference latency & memory footprint (ESP32-S3 vs STM32F407).

	ESP32-S3	STM32F407	Notes
MCU Characteristics			
Core architecture	Xtensa LX7	ARM Cortex-M4	
Clock speed (MHz)	240	168	
Compiler	xtensa-esp32s3-elf-gcc	ARM compiler v6	
Stage 1: CNN model			
Est. FLOPs	~174 KFLOPs (0.17 MFLOPs)		Using *get_flops*
Model size (KB)	22.6		After 8-bit quantization
Inference time (ms)	4.9	2.9	Average over 10,000 runs
Stage 2: ResNet model (when activated)			
Est. FLOPs	~32.6 MFLOPs		Using *get_flops*
Model size (KB)	435		After 8-bit quantization
Inference time (ms)	247	223	Average over 10,000 runs
Cascaded system (CNN-ResNet)			
Firmware total flash usage (KB)	775	692	Firmware with combined model
Peak system RAM usage (KB)	289	43.8	Approx. peak system RAM
Avg. inf. time (ADL) (ms)	12.3 – 17.3	9.6 – 14.1	
Avg. inf. time (Fall) (ms)	251.9	225.9	

CNN-only decisions. SqueezeNet is a strong alternative when a lower Stage 2 latency is preferred, as its F1-score is close to that of ResNet. However, ResNet was selected as the final architecture because it achieved the best overall F1-score and the most consistent classification performance across the reported experiments.

5.5. On-Device Benchmarking

The MCU benchmark evaluates whether the proposed cascade can meet the latency and memory constraints of practical TinyML deployment. The on-device performance of the proposed system was evaluated on the ESP32-S3 and STM32F407 MCUs. The full benchmarking results are summa-

rized in Table 4. At the component level, the Stage 1 CNN is small and fast, with about 174 KFLOPs, a quantized model size of 22.6 KB, and an inference time of 4.9 ms on the ESP32-S3 and 2.9 ms on the STM32F407. The Stage 2 ResNet is more computationally demanding, with about 32.6 MFLOPs, a quantized model size of about 435 KB, and an inference time of 247 ms on the ESP32-S3 and 223 ms on the STM32F407 when activated.

For memory usage, the combined firmware required 775 KB of flash on the ESP32-S3 and 692 KB on the STM32F407. Peak RAM usage was 289 KB on the ESP32-S3 and 43.8 KB on the STM32F407. These values indicate that the cascaded model fits within the memory budgets of the tested MCU platforms. The difference in RAM usage is mainly related to platform and toolchain behavior during model execution.

At the system level, the measured trigger rate of 1.6% at θ = 0.995 yields average inference times of about 8.9 ms on the ESP32-S3 and 6.5 ms on the STM32F407. Under the more conservative ADL scenario with a 3%–5% ResNet invocation rate, the average latency increases to 12.3–17.3 ms on the ESP32-S3 and 9.6–14.1 ms on the STM32F407. In the fall scenario, where the ResNet is invoked for every window, the latency is 251.9 ms and 225.9 ms, respectively. These values remain below the 500 ms sliding-window interval, supporting real-time on-device inference for post-event fall detection on the tested MCU platforms. For stricter latency budgets, the modular Stage 2 design allows a lighter backbone, such as SqueezeNet or MobileNetV1, to be considered with a small reduction in F1-score.

6. Discussion

The experimental results suggest that the proposed cascaded CNN-ResNet architecture is a promising direction for fall detection on MCU-based wearable devices. The model achieved high LOSO performance while keeping the average inference cost low by selectively activating the second stage. However, these findings should be interpreted within the scope of the present study. The dataset is based on simulated falls from healthy young adults, and the deployment analysis reports latency and memory usage rather than direct power measurements. Therefore, the following discussion focuses on detection performance, average computation, and deployment feasibility, while recognizing the current limits of clinical and energy-related claims.

6.1. Comparison with State-of-the-Art

The comparison with prior studies should be interpreted carefully, as the reported methods do not use the same validation protocol or even the same fall detection objective. As summarized in Table 5, some studies use hold-out validation, some use K-fold validation, and others use LOSO validation. In addition, not all methods report the same set of metrics, model size, or deployment results. For this reason, Table 5 is intended to provide context rather than a strict ranking.

Within this context, the proposed model compares favorably with several recent methods while being evaluated under LOSO validation and supported by MCU deployment results. The reported F1-score is higher than those of FDSNeXt [17], PreFallKD [10],

Table 5

Benchmark comparison with SOTA fall detection models on KFall dataset.

Method	Validation	Accuracy (%)	Precision (Fall, %)	Recall (Fall, %)	F1-Score (Fall, %)	Model size
FDSNeXt	K-fold	92.52	-	-	92.34	-[2]
PreFallKD	LOSO	98.05	90.62	94.79	92.66	0.06 MB
Ensemble ML	Hold-out	98.00	-	99.00	95.00	-[2]
TinyFallNet	Hold-out	98.00	-	99.32	98.06[1]	0.70 MB
CNN-BiLSTM	Hold-out	97.93	100	97.96	98.96[1]	1.73M param.
CNN-ResNet	**LOSO**	**98.97**	**98.48**	**97.96**	**98.21**	**0.46 MB**

[1] *F1-score was not directly reported in the summary table of the source paper;*

[2] *Model size was not reported.*

and Ensemble ML [9], and it is close to the strongest hold-out-based results. TinyFallNet is a particularly relevant reference because it targets TinyML deployment. However, TinyFallNet [25] is designed for pre-impact fall detection, whereas the present study addresses post-event or window-based fall detection on KFall. The two settings are related but not identical, so the comparison should be viewed as complementary rather than direct.

The CNN-BiLSTM [35] result is also informative. Its reported F1-score is slightly higher than that of the proposed model, but it was obtained under hold-out validation rather than LOSO validation. In addition, the network contains 1.73 million parameters, and recurrent models such as BiLSTM usually require substantial sequential computation. Since no MCU deployment metrics are reported, this architecture may be more challenging to deploy on small wearable devices than the proposed cascaded model. Taken together, these comparisons suggest that the proposed system offers a strong compromise among subject-independent evaluation, compact quantized size, and measured on-device feasibility.

6.2. Computation-Awareness Paradigm

The results also highlight the value of computation-aware inference as the main conceptual contribution of this work. The key point is not only that Stage 2 is activated selectively, but also that the system reuses compact CNN embeddings through the FIFO buffer. In many cascaded systems, uncertain inputs are passed to a stronger classifier, and intermediate features from earlier windows are discarded. In the proposed design, the embeddings extracted by Stage 1 are preserved and assembled into a short temporal context for Stage 2. This allows the ResNet to refine the decision using motion history without reprocessing raw sensor signals from scratch.

This feature reuse is particularly useful in WFDS because uncertainty often develops across a sequence of adjacent windows rather than within a single isolated window. A rapid sit-down, lying-down transition, or chair collapse may appear to be a fall when viewed in a single short segment. When the recent embedding history is taken into account, the temporal progression becomes clearer. This is consistent with the trigger analysis in Section 5.3, where routine ADLs usually remained below 1% trigger rate, while fall-adjacent segments reached much higher trigger rates. Those segments tend to contain abrupt acceleration changes, posture transitions, and impact-like motion patterns that overlap more strongly with the fall class, which naturally reduces Stage 1 confidence and activates Stage 2 more often.

Another useful implication is that computation-aware inference can reduce average latency without requiring a large model to run at every window. In this study, the benefit should be interpreted in terms of average computation and average latency rather than measured power savings, since direct power measurements were not collected. The threshold-based routing mechanism may also offer a path toward user-specific adaptation. For example, a more conservative threshold could be used for users at higher fall risk, while a stricter threshold could reduce unnecessary Stage 2 activation in low-risk settings. However, threshold personalization was not evaluated in the present study and should therefore be viewed as a potential extension for future work.

6.3. Limitations and Future Work

Several limitations should be considered when interpreting the present results. First, the KFall dataset contains simulated falls from healthy young adults rather than real falls from elderly users. Therefore, although the dataset is useful for controlled model evaluation, the results may not directly generalize to clinical elderly populations. Second, the present evaluation is based on window-level metrics. Although this is suitable for model comparison, event-level recall, event-level latency, and alarm delay were not measured, so the system's practical response behavior remains only partially characterized. Relatedly, the sensitivity of preprocessing choices, including window length, sliding step, and the fall-label percentage threshold, was not evaluated in this study. Future work should examine how these parameters affect event-level detection performance and alarm delay.

Third, the deployment study quantified latency, flash usage, and RAM usage, but it did not include direct power or energy measurements. As a result, the current evidence supports lower average computation and lower average latency, but not quantified energy savings. Fourth, the trigger mechanism relies on the softmax confidence of Stage 1. Although the threshold was carefully selected via nested LOSO valida-

tion, the confidence scores were not formally calibrated. This leaves open the question of threshold robustness when the data distribution varies across datasets, user groups, or sensor conditions.

These limitations suggest a clear research path for the future. The first priority is validation on real fall data from elderly users. The second is cross-dataset evaluation on benchmarks such as SisFall, URFALL, and MobiAct. Third, confidence calibration of the Stage 1 gatekeeper should be investigated, for example, using temperature scaling to improve the reliability of confidence scores. The fourth is event-level evaluation, including event recall and alarm delay. The fifth is energy profiling during continuous operation on the target hardware. Finally, deployment on lower-power MCUs would help determine how well the proposed computation-aware inference strategy scales under even tighter resource constraints.

7. Conclusion

This study presented a cascaded CNN-ResNet architecture for fall detection on TinyML wearable devices. The proposed framework combines a lightweight CNN gatekeeper with a ResNet refinement stage through a FIFO embedding buffer, allowing the system to allocate additional computation only to low-confidence windows. Under LOSO evaluation on the KFall dataset, the model achieved 98.97% accuracy, 98.48% precision, 97.96% recall, and 98.21% F1-score. The deployment results on the ESP32-S3 and STM32F407 showed that the proposed design can support real-time on-device inference within the 500 ms decision interval. The worst-case latency remained below 252 ms when the second stage was fully activated, while the average latency remained much lower during typical operation because the ResNet is invoked selectively. These findings indicate that the proposed cascade can maintain high detection performance while reducing average computation on MCU-based wearable platforms. At the same time, the present results should be interpreted within the scope of the dataset, which contains simulated falls from healthy young adults rather than real falls from elderly users. Future work will therefore focus on validation with real elderly fall data, cross-dataset evaluation, confidence calibration for the trigger mechanism, event-level evaluation, direct power measurements, and deployment on lower-power MCUs.

Acknowledgement

This work was supported by the Posts and Telecommunications Institute of Technology (PTIT).

References

1. Abdou, A., Mascret, Q., Gurve, D., Gosselin, B., Krishnan, S. Embedded TinyML Approach for Fall Detection in Geriatric Care. TechRxiv, 2025. https://doi.org/10.36227/techrxiv.174918043.39615584/v1
2. Afuan, L., Isnanto, R. R. A Comparative Study of Machine Learning Algorithms for Fall Detection in Technology-Based Healthcare System: Analyzing SVM, KNN, Decision Tree, Random Forest, LSTM, and CNN. E3S Web of Conferences, 2025, 605, 03051. https://doi.org/10.1051/e3sconf/202560503051
3. Al-Qaness, M. A. A., Jiang, Z., Shen, J. MKLS-Net: Multikernel Convolution LSTM and Self-Attention for Fall Detection Based on Wearable Sensors. IEEE Internet of Things Journal, 2025, 12(8), 10139-10149. https://doi.org/10.1109/JIOT.2024.3508877
4. Barshan, B., Turan, M. Ş. A Novel Heuristic Fall-Detection Algorithm Based on Double Thresholding, Fuzzy Logic, and Wearable Motion Sensor Data. IEEE Internet of Things Journal, 2023, 10(20), 17797-17812. https://doi.org/10.1109/JIOT.2023.3280060
5. Bet, P., Castro, P. C., Ponti, M. A. Fall Detection and Fall Risk Assessment in Older Person Using Wearable Sensors: A Systematic Review. International Journal of Medical Informatics, 2019, 130, 103946. https://doi.org/10.1016/j.ijmedinf.2019.08.006
6. Biswas, S. Machine Learning and Deep Learning in Human Activity Recognition and Fall Detection: Algorithms, Frameworks, and Applications for Sustainable Healthcare. Signals and Communication Technology. Cham: Springer Nature Switzerland, 2026. https://doi.org/10.1007/978-3-032-09241-0
7. Bui, V.-H., Tran, D.-N., Dao, T.-H., Tran, V.-A., Huy, P. Q., Tran, D.-T. Real-Time XGBoost-Based HAR for Elderly Monitoring Using Wearable Accelerometer and

Low-Resource Microcontroller. Proceedings of the 2025 International Conference on Advanced Technologies for Communications (ATC), 2025, 1-6. https://doi.org/10.1109/ATC67618.2025.11268637

8. Carande-Kulis, V., Stevens, J. A., Florence, C. S., Beattie, B. L., Arias, I. A Cost-Benefit Analysis of Three Older Adult Fall Prevention Interventions. Journal of Safety Research, 2015, 52, 65-70. https://doi.org/10.1016/j.jsr.2014.12.007
9. Ch. Gangadhar, P. P. Roy, R. Dinesh Kumar, J. V. Naga Ramesh, S. Ravikanth, N. Akhila. Wearable Sensor-Based Fall Detection for Elderly Care Using Ensemble Machine Learning Techniques. Measurement: Sensors, 2025, 101870. https://doi.org/10.1016/j.measen.2025.101870
10. Chi, T.-H., Liu, K.-C., Hsieh, C.-Y., Tsao, Y., Chan, C.-T. Prefallkd: Pre-Impact Fall Detection Via CNN-ViT Knowledge Distillation. Proceedings of the 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2023, 1-5. https://doi.org/10.1109/ICASSP49357.2023.10094979
11. David, R., Duke, J., Jain, A., Reddi, V. J., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., Rhodes, R., Wang, T., Warden, P. TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems. arXiv, 2021, arXiv:2010.08678. https://doi.org/10.48550/arXiv.2010.08678.
12. Delgado-Escaño, R., Castro, F. M., Cózar, J. R., Marín-Jiménez, M. J., Guil, N., Casilari, E. A Cross-Dataset Deep Learning-Based Classifier for People Fall Detection and Identification. Computer Methods and Programs in Biomedicine, 2020, 184, 105265. https://doi.org/10.1016/j.cmpb.2019.105265
13. García, E., Villar, M., Fáñez, M., Villar, J. R., De La Cal, E., Cho, S.-B. Towards Effective Detection of Elderly Falls with CNN-LSTM Neural Networks. Neurocomputing, 2022, 500, 231-240. https://doi.org/10.1016/j.neucom.2021.06.102
14. Gattani, A., Dixit, S., Patil, M., Gupta, M., Navghane, A., Hule, O., Srinivasan, K. Artificial Intelligence for Fall Detection in Older Adults: A Comprehensive Survey of Machine Learning, Deep Learning Approaches, and Future Directions. Ageing Research Reviews, 2026, 113, 102948. https://doi.org/10.1016/j.arr.2025.102948
15. Gaud, N., Rathore, M., Suman, U., Semwal, V. B. FI-BiLS: Fall Detection of Healthy Elderly Using IMU Sensor and BiLSTM Model. IEEE Sensors Journal, 2025, 25(19), 37124-37131. https://doi.org/10.1109/JSEN.2025.3602945
16. He, K., Zhang, X., Ren, S., Sun, J. Deep Residual Learning for Image Recognition. Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 770-778. https://doi.org/10.1109/CVPR.2016.90
17. Hnoohom, N., Mekruksavanich, S., Jitpattanakul, A. Pre-Impact and Impact Fall Detection Based on a Multimodal Sensor Using a Deep Residual Network. Intelligent Automation & Soft Computing, 2023, 36(3), 3371-3385. https://doi.org/10.32604/iasc.2023.036551
18. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv, 2017, arXiv:1704.04861. https://doi.org/10.48550/arXiv.1704.04861.
19. Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K. Q. Multi-Scale Dense Networks for Resource Efficient Image Classification. arXiv, 2018, arXiv:1703.09844. https://doi.org/10.48550/arXiv.1703.09844.
20. Hu, J., Cheng, F., Liu, M., Xu, X., Li, X. MicroFallNet: A Lightweight Model for Real-Time Fall Detection on Smart Wristbands. Pervasive and Mobile Computing, 2025, 109, 102046. https://doi.org/10.1016/j.pmcj.2025.102046
21. Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., Keutzer, K. SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <0.5MB Model Size. arXiv, 2016, arXiv:1602.07360. https://doi.org/10.48550/arXiv.1602.07360.
22. Jiang, Z., Al-Qaness, M. A. A., Al-Alimi, D., Ewees, A. A., Abd Elaziz, M., Dahou, A., Helmi, A. M. Fall Detection Systems for Internet of Medical Things Based on Wearable Sensors: A Review. IEEE Internet of Things Journal, 2024, 11(21), 34797-34810. https://doi.org/10.1109/JIOT.2024.3421336
23. Kallimani, R., Pai, K., Raghuwanshi, P., Iyer, S., López, O. L. A. TinyML: Tools, Applications, Challenges, and Future Research Directions. Multimedia Tools and Applications, 2023, 83(10), 29015-29045. https://doi.org/10.1007/s11042-023-16740-9
24. Kausar, F., Mesbah, M., Iqbal, W., Ahmad, A., Sayyed, I. Fall Detection in the Elderly Using Different Machine Learning Algorithms with Optimal Window Size. Mobile Networks and Applications, 2024, 29(2), 413-423. https://doi.org/10.1007/s11036-023-02215-6

25. Koo, B., Yu, X., Lee, S., Yang, S., Kim, D., Xiong, S., Kim, Y. TinyFallNet: A Lightweight Pre-Impact Fall Detection Model. Sensors, 2023, 23(20), 8459. https://doi.org/10.3390/s23208459

26. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE, 1998, 86(11), 2278-2324. https://doi.org/10.1109/5.726791

27. Li, Y., Liu, P., Fang, Y., Wu, X., Xie, Y., Xu, Z., Ren, H., Jing, F. A Decade of Progress in Wearable Sensors for Fall Detection (2015-2024): A Network-Based Visualization Review. Sensors, 2025, 25(7), 2205. https://doi.org/10.3390/s25072205

28. Luo, Z., Jia, S., Niu, H., Zhao, Y., Zeng, X., Dong, G. Elderly Fall Detection Algorithm Based on Improved YOLOv5s. Information Technology and Control, 2024, 53(2), 601-618. https://doi.org/10.5755/j01.itc.53.2.36336

29. Mohammed, M. A., Abd Ghani, M. K., Lakhan, A., AL-Attar, B., Khaled, W. Federated Learning-Driven IoT and Edge Cloud Networks for Smart Wheelchair Systems in Assistive Robotics. Iraqi Journal for Computer Science and Mathematics, 2025, 6(1). https://doi.org/10.52866/2788-7421.1241

30. Mohammed, M. A., Abd Ghani, M. K., Memon, S., Lakhan, A., Marhoon, H. A., Radhi, A. D., Danys, L., Martinek, R. Secure IoMT Smartwatch-Based Blood Glucose Monitoring Using Multimodal Activity and Nutrition Data with Transfer Learning. Scientific Reports, 2026, 16(1), 6736. https://doi.org/10.1038/s41598-026-35419-3

31. Al Nahian, M. J., Raisa, J. F., Mahmud, M., Kaiser, M. S., Rahman Ahad, M. A., Ghosh, T., Al Banna, M. H., Hossain, M. S., Andersson, K. Artificial Intelligence for Elderly Fall Detection: State-of-the-Art Methods, Applications and Challenges. Cognitive Computation, 2026, 18(1), 12. https://doi.org/10.1007/s12559-026-10550-5

32. Naja, S., Din Makhlouf, M. M. E., Chehab, M. A. H. An Ageing World of the 21st Century: A Literature Review. International Journal of Community Medicine and Public Health, 2017, 4(12), 4363. https://doi.org/10.18203/2394-6040.ijcmph20175306

33. Putra, I. P. E. S., Vesilo, R. Window-Size Impact on Detection Rate of Wearable-Sensor-Based Fall Detection Using Supervised Machine Learning. Proceedings of the 2017 IEEE Life Sciences Conference (LSC), 2017, 21-26. https://doi.org/10.1109/LSC.2017.8268134

34. Rahmath P, H., Srivastava, V., Chaurasia, K., Pacheco, R. G., Couto, R. S. Early-Exit Deep Neural Network - A Comprehensive Survey. ACM Computing Surveys, 2025, 57(3), 1-37. https://doi.org/10.1145/3698767

35. Sahni, S., Jain, S., Saritha, S. K. A Novel Ensemble Model for Fall Detection: Leveraging CNN and BiLSTM with Channel and Temporal Attention. Automatika, 2025, 66(2), 103-116. https://doi.org/10.1080/00051144.2025.2450553

36. Saleh, M., Le Bouquin Jeannes, R. An Efficient Machine Learning-Based Fall Detection Algorithm Using Local Binary Features. Proceedings of the 2018 26th European Signal Processing Conference (EUSIPCO), 2018, 667-671. https://doi.org/10.23919/EUSIPCO.2018.8553340

37. Sethuraman, S. C., Kompally, P., Mohanty, S. P., Choppali, U. MyWear: A Novel Smart Garment for Automatic Continuous Vital Monitoring. IEEE Transactions on Consumer Electronics, 2021, 67(3), 214-222. https://doi.org/10.1109/TCE.2021.3085888

38. Sözer, A. T. A Cost-Effective Embeddable Algorithm for Accelerometer-Based Fall Detector. Engineering Science and Technology, an International Journal, 2025, 102185. https://doi.org/10.1016/j.jestch.2025.102185

39. Suardiyana Putra, I. P. E., Brusey, J., Gaura, E. A Cascade-Classifier Approach for Fall Detection. EAI Endorsed Transactions on Pervasive Health and Technology, 2015, 2(8), e2. https://doi.org/10.4108/eai.14-10-2015.2261619

40. Teerapittayanon, S., McDanel, B., Kung, H. T. BranchyNet: Fast Inference Via Early Exiting from Deep Neural Networks. Proceedings of the 2016 23rd International Conference on Pattern Recognition (ICPR), 2016, 2464-2469. https://doi.org/10.1109/ICPR.2016.7900006

41. Tinetti, M. E., Speechley, M., Ginter, S. F. Risk Factors for Falls among Elderly Persons Living in the Community. New England Journal of Medicine, 1988, 319(26), 1701-1707. https://doi.org/10.1056/NEJM198812293192604

42. Verma, S. K., Willetts, J. L., Corns, H. L., Marucci-Wellman, H. R., Lombardi, D. A., Courtney, T. K. Falls and Fall-Related Injuries among Community-Dwelling Adults in the United States. PLOS ONE, 2016, 11(3), e0150939. https://doi.org/10.1371/journal.pone.0150939

43. Villa, M., Casilari, E. Energy-Efficient Fall-Detection System Using LoRa and Hybrid Algorithms. Biomi-

metics, 2025, 10(5), 313. https://doi.org/10.3390/biomimetics10050313

44. World Health Organization. Ageing and Health. World Health Organization, 2024. Available: https://www.who.int/news-room/fact-sheets/detail/ageing-and-health.
45. World Health Organization. Step Safely: Strategies for Preventing and Managing Falls across the Life-Course. World Health Organization, 2021, 194 pp. Available: https://www.who.int/publications/i/item/978924002191-4.
46. Yahyati, C., Lamaakal, I., Maleh, Y., El Makkaoui, K., Ouahbi, I., Almousa, M. A Systematic Review of State-of-the-Art TinyML Applications in Healthcare, Education, and Transportation. IEEE Access, 2025, 13, 204513-204562. https://doi.org/10.1109/ACCESS.2025.3633575
47. Yu, X., Park, S., Kim, D., Kim, E., Kim, J., Kim, W., An, Y., Xiong, S. A Practical Wearable Fall Detection System Based on Tiny Convolutional Neural Networks. Biomedical Signal Processing and Control, 2023, 86, 105325. https://doi.org/10.1016/j.bspc.2023.105325
48. Yu, X., Jang, J., Xiong, S. A Large-Scale Open Motion Dataset (KFall) and Benchmark Algorithms for Detecting Pre-Impact Fall of the Elderly Using Wearable Inertial Sensors. Frontiers in Aging Neuroscience, 2021, 13, 692865. https://doi.org/10.3389/fnagi.2021.692865
49. Zaid Salah, O., Selvaperumal, S. K., Abdulla, R. Accelerometer-Based Elderly Fall Detection System Using Edge Artificial Intelligence Architecture. International Journal of Electrical and Computer Engineering, 2022, 12(4), 4430-4438. https://doi.org/10.11591/ijece.v12i4.pp4430-4438

This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution 4.0 (CC BY 4.0) License (http://creativecommons.org/licenses/by/4.0/).