A Lightweight CNN for Real-Time Pre-Impact Fall Detection

Cristian Turetta*, Muhammad Toqeer Ali*, Florenc Demrozi†,and Graziano Pravadelli‡
*Dept. of Computer Science, University of Verona, Italy, name.surname@univr.it
†Dept. of Electrical Engineering and Computer Science, University of Stavanger, Norway, florenc.demrozi@uis.no
‡Dept. of Engineering for Innovation Medicine, University of Verona, Italy, graziano.pravadelli@univr.it

***Abstract*—Falls can have significant and far-reaching effects on various groups, particularly the elderly, workers, and the general population. These effects can impact both physical and psychological well-being, leading to long-term health problems, reduced productivity, and a decreased quality of life. Numerous fall detection systems have been developed to prompt first aid in the event of a fall and reduce its impact on people's lives. However, detecting a fall after it has occurred is insufficient to mitigate its consequences, such as trauma. These effects can be further minimized by activating safety systems (e.g., wearable airbags) during the fall itself—specifically in the pre-impact phase—to reduce the severity of the impact when hitting the ground. Achieving this, however, requires recognizing the fall early enough to provide the necessary time for the safety system to become fully operational before impact. To address this challenge, this paper introduces a novel lightweight convolutional neural network (CNN) designed to detect pre-impact falls. The proposed model overcomes the limitations of current solutions regarding deployability on resource-constrained embedded devices, specifically for controlling the inflation of an airbag jacket. We extensively tested and compared our model, deployed on an STM32F722 microcontroller, against state-of-the-art approaches using two different datasets.**

***Index Terms*—Fall detection, pre-impact detection, safety devices, airbag jackets, inertial sensors**

I. Introduction

Falls represent a major public health concern due to their profound impact on various people, particularly the elderly and the workers in risk situations (e.g., at worksites). The consequences of falls extend beyond physical injuries, leading to psychological distress, reduced quality of life, and a significant burden on healthcare systems. In the USA, fall-related accidents accounted for 16% of the total non-fatal occupational injuries [1]. Besides this, falls are a cause of death in all sectors, responsible for 865 out of 5,486 total fatalities in 2022 [1]. While fall prevention is the ideal solution, there is a need to improve the effectiveness of solutions to mitigate the effect of falls when they irremediably happen.

This study was supported by the Protechto s.r.l. company, and by the following projects: PREPARE under Grant n. F/310130/05/X56, Grant CUP: B39J23001730005, and Grant D.M. MiSE 31/12/2021, UNISCO under the (FSE)+ 2021-2027 Grant CUP: 1695-0026-553-2023, and CollaborICE under the MICS (Made in Italy – Circular and Sustainable) Extended Partnership from Next-Generation EU (Italian PNRR – M4 C2, Invest 1.3 – D.D. 1551.11-10-2022, PE00000004), CUP MICS D43C22003120001 - Cascade funding project CollaborICE.

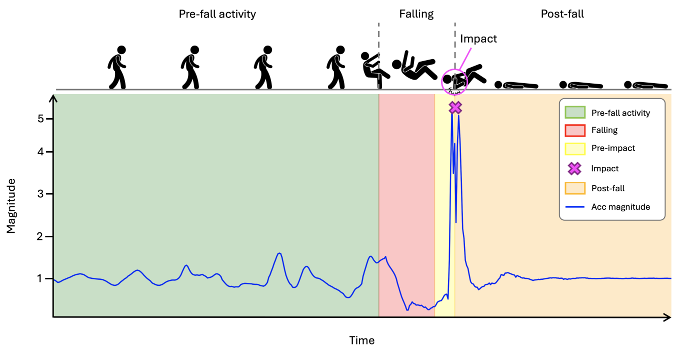

Fig. 1: Fall stages: pre-fall activity phase in green, falling phase in red and yellow, where the yellow area indicates the last 150 ms before impact (marked by a violet cross), and post-fall phase in orange.

As depicted in Figure 1, a fall event comprises four different stages: **pre-fall**, **falling** (aka., pre-impact), **impact**, and **post-fall** [2]. The pre-fall stage consists of typical human activities (e.g., walking, climbing stairs, jumping, etc.) taking place before a fall-inducing event, such as tripping, slipping, or some other disturbance. Once such an event occurs, the individual enters a state of free fall, from which recovery to a stable position is nearly impossible. The impact phase marks the end of the fall and occurs when the body makes contact with the ground. Concluding the cycle, in the post-fall phase, the individual typically remains motionless on the floor for a non-specified amount of time.

State-of-the-art fall detection systems primarily focus on detecting the fall (i.e., the impact event) or the post-fall phase (i.e., lying on the floor) [3], [4]. These systems enable post-event alerting mechanisms designed to provide prompt medical assistance, thereby reducing the time a person remains on the floor after a fall. However, these approaches make it impossible to prevent injuries caused by the fall, thus implementing a reactive rather than a proactive approach.

In the proactive approach direction, researchers have proposed various falling detection techniques to mitigate the health impact of falls, ranging from early threshold-based methods to more advanced machine learning (ML) and deep learning (DL) solutions, all aimed at accelerating human or assistive/protection interventions. Threshold-based techniques

Ref.	Year	Model	Accuracy	Recall	F1-Score	WS (ms)	Dataset	# Subjects	# ADLs / # Falls
[5]	2024	CNN-BiGRU	98.00	98.00	98.00	900	[6]	32	21/15
[7]	2023	PreFallKD (CNN)	98.05	94.79	92.66	500	[6]	32	21/15
[8]	2022	FallNet (LSTM-CNN)	97.52	99.24	98.79	1000	[6], [9]	32,38	7/6, 7/6
[6]	2021	ConvLSTM	99.16	99.32	99.01	500	[6]	32	21/15
[10]	2021	Threshold algorithm	95.86	94.04	97.67	-	[9], Own	38, 9	10/5, 86/135
[11]	2020	Threshold algorithm	92.40	96.10	94.20	-	[9]	30	14/06

TABLE I: Recent related work on pre-impact fall detection.

rely on accelerometer and gyroscope data perceived by an inertial measurement unit (IMU) worn on different parts of the body to identify an anomaly in the movement attributable to a fall or falling start event [6], [10]–[12]. They are typically faster than ML/DL methods since they analyze a smaller amount of data in a shorter time frame (e.g., a sequence of three accelerometer samples covering 30 ms at 100 Hz sampling rate) to detect movement anomalies; however, they generally achieve a lower accuracy. ML and DL approaches [5], [7], [13], [14] offer higher accuracy, but they come with increased computational demands and hardware requirements. In fact, they require the analysis of a larger dataset than threshold-based solutions (e.g., 50 accelerometer samples covering a 500 ms time frame) to make their predictions. Furthermore, a significant challenge in recognizing a falling event is that the falling time frame (i.e., from the fall start to the impact) is less than 500 ms long in 50% of fall events. This leaves minimal intervention time, including the time required for the fall detection algorithm to recognize the falling, trigger the deployment of a life-saving device (such as a wearable airbag), and allow the device to fully operate. As a result, the available time for effective intervention is often close to zero.

Table I summarizes the recent falling detection studies in the literature. Kiran *et al.* [5] proposed a CNN-BiGRU model capable of achieving an average accuracy and F1-score of 98% on the KFall dataset [6]. Similarly, [7] proposed a CNN-Vision Transformer knowledge distillation, named PreFallKD, capable of achieving an F1-score of 92.66% on the KFall dataset. Other researchers proposed LSTM-based approaches [6], [8], [15], but such a model can hardly be implemented on resource-constraint devices.

Based on the mentioned challenges (i.e., computational and time requirements), the existing ML/DL state-of-the-art approaches share the following real-world utilization limitations: i) the size of the data window used as input is considerable (see WS column in Table I), enabling only the recognition of long-duration falls (from 500 ms and above), ii) they have been trained by considering the whole duration of the falling phase, but in the real conditions, the last 150 ms of the falling phase cannot be used, as this is generally the time required to activate a safety device (e.g., an airbag jacket); iii) there is a lack of analysis concerning their deployability on resource-constrained devices to drive fall-mitigation equipment. In addition, we encountered a lack of clarity regarding their training and validation procedures (e.g., whether they use *k*-fold cross-validation, basic train-test split, subject-dependent, or subject-independent approaches). In some cases, they use similar data (i.e., from the same subjects and data collection sessions) for training and testing, compromising the generalization capabilities of the approaches.

To address these limitations, we propose a lightweight CNN model scrupulously designed through a subject-independent cross-validation approach for the training and testing phases, ensuring a robust evaluation of its performance. In addition, we applied lightweight preprocessing techniques to the inertial data, such as low-pass filtering and segmentation, making it suitable for real-time deployment on a resource-constrained device (i.e., STM32F722 microcontroller). We used various data augmentation techniques to mitigate data imbalance (as only 1-3% of the data in fall datasets concerns falls). Furthermore, we thoroughly analyzed different window sizes for feeding the CNN (200 ms, 300 ms, and 400 ms) to identify the optimal configuration, aiming for the best performance with shorter windows. This is crucial to provide a timely trigger signal to activate safety systems and then protect the individual before impact with the ground. For the same reason, unlike the state-of-the-art solutions, we did not provide the final 150 ms of each falling event to train our CNN. This means the recognition of the falling for our CNN is a more difficult task than for existing methods, but it guarantees that it will be more reactive in triggering the safety system. We compared our approach against state-of-the-art solutions by using two different datasets, the freely-available KFall dataset [6], and a self-collected dataset, which consists of, respectively, 32 and 29 participants, 15 and 21 distinct types of falls, and 21 and 23 daily activities.

Next sections are organized as follows. Section II provides information on the datasets. Section III exposes the proposed approach. Section IV showcases the experimental results. Section V provides the concluding remarks and future work.

II. Preliminaries on Fall Datasets

As previously mentioned, we utilized two different datasets in the design of this methodology: a) the KFall, as proposed in [6], which is a widely-adopted dataset for fall detection during activities of daily life (ADLs), and b) a self-collected dataset which extends KFall with additional activities and falls types specific to construction site environments (e.g., backward/forward falls from height). The rest of this section is dedicated to describing the characteristics of the self-collected dataset, while for KFall, the reader is referred to [6].

A. Data Acquisition System

In constructing the self-collected dataset, we defined three relevant requirements: i) synchronization between video and sensor data to precisely label the falling events, ii) recording of a variety of ADLs and falls (including falls related to high-risk environments) from a large set of different subjects, iii) adoption of a high sampling frequency and precise inertial sensor measurements.

The inertial data included in the dataset was acquired by using a custom printed circuit board (PCB), produced by the Protechto company, powered by an STM32F722RET6 microcontroller, and equipped with an ARM Cortex-M7 32-bit RISC core operating up to 216 MHz with advanced peripherals, including an FPU, suitable for real-time processing. The PCB features an LIS3DH tri-axis MEMS accelerometer (±2g to ±16g, data rates up to 5 kHz, and resolution of 1 mg). Embedded in the rear of a safety jacket provided by Protechto[1], the sensor module measures tri-axial acceleration and angular velocity. Three LEDs mounted in the front of the safety jacket are used to synchronize the start of data collection with video recording. At each time instant (every 10 ms, given the 100 Hz sampling frequency), we collected the accelerometer and the gyroscope data, and we computed on the edge the Eulerian angle data (pitch, roll, yaw) to capture detailed movement dynamics.

B. Participants and Activities

Our dataset comprises 29 participants (24 males, 5 females) in healthy conditions, and none present mobility impairments or musculoskeletal disorders. To participate in the data collection session, every participant signed an informed consent for the experimental protocol. The participants' average age is 23.5 years (±6.3 years), with average height and weight of 71.5 Kg (±13.2 Kg) and 178 cm (±8 cm), respectively.

The dataset contains 23 types of ADLs and 21 types of falls for each subject, included those of the KFall dataset. For the ADLs, the dataset covers from simple daily movements (e.g., walking, sit-to-stand, etc.) to dynamic activities (e.g., jogging, jumping) and even near-fall scenarios (e.g., stumbling during walking, collapsing to a chair, going down a ladder, etc.). On the other hand, the fall macro-categories are the following: i) falls from walking (e.g., caused by a slip or a trip), ii) falls from sitting (e.g., caused by fainting or trying to get up), iii) falls from standing (e.g., trying to sit down), iv) falls from high (e.g., from a ladder or scaffold). All the activities, with their identification code, are reported in Table II. In red, activities concluded with a fall; in green, ADLs without falls.

Task	Description
01	Stand for 30 seconds
02	Stand, slowly bend, tie shoe lace, and get up
03	Pick up an object from the floor
04	Gently jump (try to reach an object)
05	Stand, sit to the ground, wait a moment, and get up with normal speed
06	Walk normally with turn
07	Walk quickly with turn
08	Jog normally with turn
09	Jog quickly with turn
10	Stumble with obstacle while walking
11	Sit on a chair for 30 seconds
12	Walk downstairs normally
13	Sit down to a chair normally, and get up from a chair normally
14	Sit down to a chair quickly, and get up from a chair quickly
15	Sit a moment, trying to get up, and collapse into a chair
16	Walk downstairs quickly
17	Lie on the floor for 30 seconds
18	Sit a moment, lie down to the floor normally, and get up normally
19	Sit a moment, lie down to the floor quickly, and get up quickly
20	Forward fall when trying to sit down
21	Backward fall when trying to sit down
22	Lateral fall when trying to sit down
23	Forward fall when trying to get up
24	Lateral fall when trying to get up
25	Forward fall while sitting, caused by fainting
26	Lateral fall while sitting, caused by fainting
27	Backward fall while sitting, caused by fainting
28	Vertical (forward) fall while walking caused by fainting
29	Fall while walking, use of hands to dampen fall, caused by fainting
30	Forward fall while walking caused by a trip
31	Forward fall while jogging caused by a trip
32	Forward fall while walking caused by a slip
33	Lateral fall while walking caused by a slip
34	Backward fall while walking caused by a slip
35	Walk upstairs normally
36	Walk upstairs quickly
37	Backward fall while slowly moving back
38	Backward fall while quickly moving back
39	Forward fall from height
40	Backward fall from height
41	Backward fall while trying to climb up the ladder
42	Backward fall while trying to climb down the ladder
43	Climb up and climb down the stairs
44	Walk slowly and jump over the obstacle

TABLE II: Activities performed in our dataset.

C. Data Labeling

The sensor's data and the smartphone's video, collected at the same frequency (100 Hz/fps), are synchronized using the LED lights on the front part of the safe jacket. When the LEDs turn red, the board starts the data collection, thus indicating the synchronization point between the video and the sensor. Through this synchronization, the falling is annotated with a frame-by-frame approach. The falling starts in the frame in which the subject cannot recover from the free fall status, while the impact moment occurs in the frame where the safety jacket touched the ground.

[1]Equipped with an airbag that requires 150 ms, starting from the triggering event, to reach the full extension.

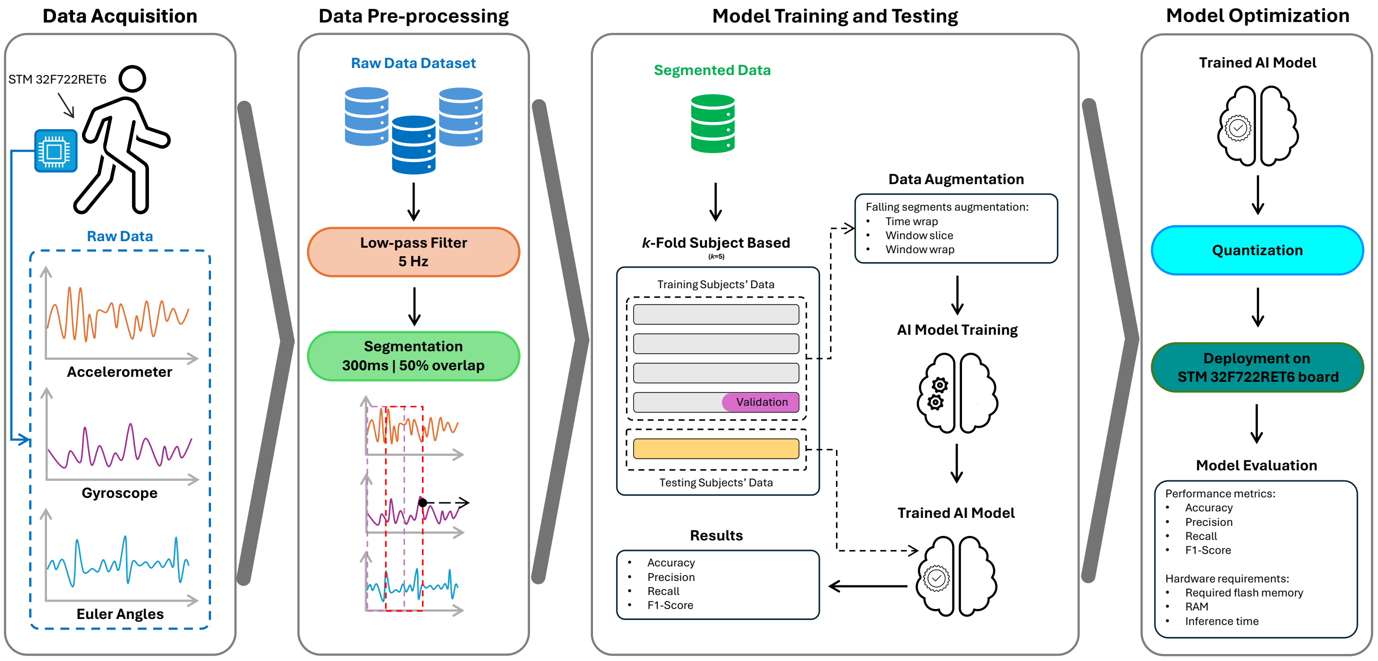

Fig. 2: Schematic representation of the methodological approach.

III. Methodology

This section outlines the methodology for recognizing the pre-impact (i.e., falling) events. Figure 2 provides a detailed view of the methodology that started with the data acquisition as introduced in Section II. Identifying pre-impact events on edge devices is challenging due to the complexity of human movements, the short duration of falls (ranging, generally, from 150 ms to 1100 ms), and the limited computational resources available on edge devices. Therefore, in designing our methodology, we focused on using computationally efficient processes to meet these constraints.

A. Data Pre-processing

In this phase, we applied two lightweight data preprocessing steps. First, we used a fourth-order Butterworth low-pass filter (5 Hz) on the raw data to eliminate noise. Second, we segmented the filtered data, allowing smooth transitions between segments. To optimize performance, we experimented with different segment sizes (ranging from 100 ms to 400 ms) and various overlap sizes (from 0% to 75%, in increments of 25%). Given the objective of designing an accurate system for both daily environments and workplaces, where participants perform irregular movements in highly-noisy settings, the noise removal step is critically important. The segment is represented as a matrix composed of n row representing the number of snapshots (i.e., obtained every 10 ms) and m columns related to the number of features. In our setup, we fixed $m = 9$, and the features are accelerometer and gyroscope (x, y, z, for each) and Eulerian angles ($pich$, $roll$, yaw). For example, if $n = 20$ corresponds to 200 ms, the segment (i.e., matrix) size will be $[20 \times 9]$.

B. Model Architecture

The goal of our method is to use the sensor features processed as described in the previous section to detect falling in real conditions before the subject impacts the ground. To achieve this, we propose a memory- and computation-efficient deep learning solution implemented as a lightweight CNN.

The CNN model's architecture splits the input matrix into three matrices, each with dimension $n \times 3$, thus splitting the three motion features (i.e., accelerometer, gyroscope, and Eulerian angles) in each matrix. Each motion feature's matrix passes through a convolutional layer and then a max pooling layer to extract characteristics from the data. Subsequently, these three branches' outputs are concatenated together and then fed to two dense layers. The two dense layers comprise 64 and 32 neurons, respectively, and their activation function is the rectified linear unit (ReLu). Since we have two classes to be recognized the model's output is a dense layer activated by a sigmoid function, corresponding to *activity* and *falling*, respectively. The output of the sigmoid function is bounded between zero and one. Thus, the prediction p can be any value between zero and one, indicating the confidence in that sample being positive.

C. Model Training and Testing

To ensure a thorough and robust evaluation of the proposed deep learning model, we employed a subject-based k-fold cross-validation technique (k = 5) using a combined dataset including the self-collected and the KFall data, resulting in a total of 61 participants. In each iteration of the k-fold cross-validation, one fold (containing data from 12 subjects) is used for testing, while the remaining four folds are used for training.

Additionally, four randomly selected subjects from the training set (not used for training) are used for model validation.

This cross-validation methodology guarantees no overlap between the training/validation and testing data, as they involve different subjects. This setup simulates a real-world scenario where the fall detection model is trained on data from one group of individuals and applied to new, unseen users.

Additionally, we applied two data augmentation techniques, specifically *time warping* and its *window warping* variant, to the training data corresponding to the fall segments of the time series. The time warping stretches and compresses the signal, making the model more robust to the sampling of the signal [16], while window warping warps a randomly selected falling time series segment by speeding it up or down [17]. In this way, we simulated the variation in the overall falling event's speed, thus producing more/less segments labeled as falling w.r.t the original data.

Even though the data augmentation helps to provide more instances of the minority class, the training dataset is still unbalanced. Thus, to address this problem during the training, the classes we want to classify (activities and falling) have different weights. Furthermore, we set a bias on the output layer of the models. The bias of each class is calculated on the ratio of the positive and negative class instances. The formula for calculating the initial bias for a class is the following:

$$b_i = \log\left(\frac{p_i}{1-p_i}\right) \tag{1}$$

where p_i is the prior probability of class i, calculated as:

$$p_i = \frac{\text{number of instances in class } i}{\text{total number of instances}} \tag{2}$$

The number of epochs for the model training is 200. However, we used early stopping with a patience of 20 epochs based on the validation loss, thus restoring the model's weights to those obtained in the best epoch.

At the end of the training, the model is tested on the testing subject's data, as depicted in Figure 2, and evaluated in terms of Accuracy, Precision, Recall, and F1-Score.

D. Model Optimization

As we intend to use the fall detection model to drive the activation of a safety device, we have to guarantee its deployability in a resource-constraint embedded controller. Deploying a deep learning model on a resource-constrained device requires a quantization step. Then, we applied post-training quantization, which converts the model's floating-point weights and activations to a lower precision format. Specifically, we adopted 8-bit integer conversion after the training phase. By reducing the precision of numerical representations, post-training quantization significantly decreases both the model size and its computational requirements. The CNN model has been successfully deployed on an STM32F722 microcontroller.

IV. Experimental Results

This section first outlines the data preparation process used to align the KFall and the self-collected datasets, then it evaluates the effectiveness of our pre-impact fall detection algorithm, and finally it shows its performance at the edge.

A. Dataset Alignment

To develop an algorithm with strong generalization capabilities, we merged our proposed dataset with the KFall dataset [6], thereby increasing the number of subjects and the volume of data for training and testing. Moreover, since both datasets use identical sensor placements but not orientation, it was necessary to align the sensor orientations of the KFall dataset with our own to ensure consistency. We achieved this alignment using a rotation matrix computed through Rodrigues' rotation formula, which adjusted the original sensor orientation of the KFall dataset to match ours. Additionally, we standardized the units of measurement across both datasets, converting all values to gravitational acceleration (g) for uniformity. This process ensures compatibility between the datasets and fulfills the secondary objective of increasing the dataset size, contributing to enhanced model training and improved generalization capabilities.

B. Evaluation of the Pre-Impact Recognition Model

To demonstrate the effectiveness of the proposed methodology, we tested the approach under various segment sizes and overlap configurations. Additionally, we compared the results of our method with ML and DL models used as baselines, such as Multi-Layer Perceptrons (MLP), as well as models commonly employed at the state-of-the-art, including Long Short-Term Memory networks (LSTM) and ConvLSTM2D, similarly to what adopted in [6], [8]. The results are presented in Table III.

The proposed CNN model (fourth row in Table III) outperforms the other models in terms of accuracy, precision, recall, and F1-score across all tested segment sizes. The best results were achieved with a segment size of 400 ms and 50% of overlapping. Despite this significant class imbalance, with a total number of 242,738 (95.4%) non-falling segments and 9,105 (3.6%) falling segments, the model shows a strong ability to correctly identify falling segments while avoiding a bias toward the majority class (ADLs without falls).

It is worth noting that these results should not be negatively compared with those of Table I, since the methods included in Table I do not remove the last 150 ms of the falling phase from the training dataset, which is the most characterizing set of data for detecting a fall, resulting in a simpler classification task than ours. Thus, while their F1 score (from $\sim$ 92% to $\sim$ 99%) appears to be better than our ($\sim$ 87%), they cannot effectively drive a safety device, as their trigger signal may arrive too late to allow it becomes fully operational before the impact.

In addition, a deeper analysis must be conducted to evaluate the effectiveness of our approach. Measuring the model's performance just by looking at the number of correctly classified falling and non-falling segments, as done by the approaches in Table I, is misleading. A falling/non-falling event is composed of several segments. While a certain number

Model	200 ms segment size (100 ms overlap)				300 ms segment size (150 ms overlap)				400 ms segment size (200 ms overlap)			
	Accuracy	*Precision*	*Recall*	*F1-Score*	*Accuracy*	*Precision*	*Recall*	*F1-Score*	*Accuracy*	*Precision*	*Recall*	*F1-Score*
MLP	96.76	51.24	50.00	49.18	96.62	53.02	55.39	54.13	96.45	60.23	54.63	54.25
LSTM	97.28	80.92	68.62	72.98	97.43	82.51	72.08	75.93	97.60	85.97	75.74	79.81
ConvLSTM2D	97.12	81.24	61.61	66.37	97.21	83.67	63.55	68.53	97.10	85.57	65.36	70.75
CNN (Proposed)	97.93	85.61	78.85	81.75	98.01	86.38	80.03	82.85	98.28	90.40	83.95	86.69

TABLE III: Comparison of the tested models on different segment sizes. Results are expressed in %.

of segments belonging to a falling event could be misclassified as non-falling, it is enough to correctly classify one segment to effectively predict the fall (and then activate the safety system). Similarly, a single misclassification of a segment belonging to a non-falling event (i.e., an ADL) may cause the useless activation of the safety system. Thus, the performance of a pre-impact classifier must be analyzed at the event level rather than at the segment level. Table IV reports the results of this analysis. From the observation of Table IVa, we can derive that, on average, only 4.17% of fall events are misclassified as ADLs. More importantly, on average, we have only 2.04% of false positives (right part of the table), i.e., ADLs misclassified as falls. A low number of false positives is of utmost importance if the model is used to drive a safety device; in fact, unnecessary activations make it impractical. Then, we configured our model to minimize false positives, even at the cost of missing the detection of some actual falls. Finally, a consideration is also necessary to highlight that in Table IVb, the ADLs with the highest percentage of misclassification (red ADLs) correspond to activities that workers rarely perform in risky situations (e.g., at construction sites) or by elderly, two categories of individuals that would benefit from wearing safety devices driven by our model.

C. On-Edge Performance

The CNN was developed using the TensorFlow and Keras frameworks. After optimization, the model's size is 67.03 KiB, with a total RAM usage of 16.87 KiB. Importantly, the model's performance remains unchanged after quantization, consistent with the best model results presented in Table III and Table IV. The compact design of the CNN model allowed us to deploy it on a resource-constrained device, i.e., a custom board powered by an STM32F722RET6 microcontroller with 256 KiB of flash memory and RAM. The inference time for each segment computed on the board is 4 ms $\pm$ 3 ms, with 4 ms accounting for the CNN model's inference and the additional 3 ms for the sensor data fusion phase that occurs prior to the model's execution.

V. Conclusions

In this paper, we proposed a pre-impact fall detection method by designing a lightweight CNN model that can be deployed on a resource-constrained device to drive the activation of safety systems like airbag jackets. The primary objective of the method is to recognize the onset of a fall as early as possible to enable the activation of the safety aid before the impact with the ground (to this aim we truncated fall events by 150 ms to account for the time generally needed by a wearable airbag to inflate); the second is to avoid useless and discomforting activations. Through an exhaustive subject-independent cross-validation approach, the model demonstrated (i) robust performance to ensure the model's effectiveness in recognizing falling events by using shorter segments of data ($\leq$ 400 ms) than state-of-the-art approaches and (ii) a very high precision in avoiding false positives, particularly in case of ADLs carried out by elderly and workers moving in risky environments.

Task ID	Miss %
39	16.00%
40	12.00%
21	9.47%
22	8.42%
41	8.00%
33	6.95%
27	5.35%
29	4.42%
37	4.00%
42	4.00%
30	3.85%
31	3.37%
32	3.17%
28	2.73%
34	2.72%
26	2.19%
23	2.17%
24	1.61%
25	1.60%
20	1.60%
38	0.00%
All actions	**4.17%**

(a) Falls misclassified as ADLs

Task ID	Miss %
44	20.00%
15	11.29%
19	6.74%
4	6.35%
5	2.16%
10	2.13%
14	1.63%
8	1.62%
18	1.10%
9	0.56%
16	0.56%
3	0.54%
1	0.00%
2	0.00%
6	0.00%
7	0.00%
11	0.00%
12	0.00%
13	0.00%
17	0.00%
35	0.00%
36	0.00%
43	0.00%
All actions	**2.04%**
Red actions	**3.34%**
Green actions	**0.46%**

(b) ADLs misclassified as falls

TABLE IV: Misclassification statistics (400 ms segment size). Red ADLs are unconventional for people at risk of falls due to motor impairments (e.g., elderly) or moving in risky places (e.g., at construction sites), while green ones occur more frequently. For the descriptions of the tasks see Table II.

References

[1] U.S. Bureau of Labor Statistics, "Injuries, illnesses, and fatalities," 2024, accessed: 2024-08-20.

[2] J. Liu, X. Li, S. Huang, R. Chao, Z. Cao, S. Wang, A. Wang, and L. Liu, "A review of wearable sensors based fall-related recognition systems," *Engineering Applications of Artificial Intelligence*, vol. 121, p. 105993, 2023.

[3] A. Ramachandran and A. Karuppiah, "A survey on recent advances in wearable fall detection systems," *BioMed research international*, vol. 2020, no. 1, p. 2167160, 2020.

[4] Z. Jiang, M. A. Al-qaness, A.-A. Dalal, A. A. Ewess, M. Abd Elaziz, A. Dahou, and A. M. Helmi, "Fall detection systems for internet of medical things based on wearable sensors: A review," *IEEE Internet of Things Journal*, 2024.

[5] S. Kiran, Q. Riaz, M. Hussain, M. Zeeshan, and B. Krüger, "Unveiling fall origins: Leveraging wearable sensors to detect pre-impact fall causes," *IEEE Sensors Journal*, 2024.

[6] 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," *Frontiers in Aging Neuroscience*, vol. 13, p. 692865, 2021.

[7] T.-H. Chi, K.-C. Liu, C.-Y. Hsieh, Y. Tsao, and C.-T. Chan, "Prefallkd: Pre-impact fall detection via cnn-vit knowledge distillation," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[8] R. Jain and V. B. Semwal, "A novel feature extraction method for preimpact fall detection system using deep learning and wearable sensors," *IEEE Sensors Journal*, vol. 22, no. 23, pp. 22 943–22 951, 2022.

[9] A. Sucerquia, J. D. López, and J. F. Vargas-Bonilla, "Sisfall: A fall and movement dataset," *Sensors*, vol. 17, no. 1, p. 198, 2017.

[10] F. A. S. F. de Sousa, C. Escriba, E. G. A. Bravo, V. Brossa, J.-Y. Fourniols, and C. Rossi, "Wearable pre-impact fall detection system based on 3d accelerometer and subject's height," *IEEE Sensors Journal*, vol. 22, no. 2, pp. 1738–1745, 2021.

[11] H. Jung, B. Koo, J. Kim, T. Kim, Y. Nam, and Y. Kim, "Enhanced algorithm for the detection of preimpact fall for wearable airbags," *Sensors*, vol. 20, no. 5, 2020.

[12] S. N. Moutsis, K. A. Tsintotas, and A. Gasteratos, "Pipto: precise inertial-based pipeline for threshold-based fall detection using three-axis accelerometers," *Sensors*, vol. 23, no. 18, p. 7951, 2023.

[13] N. De Raeve, A. Shahid, M. De Schepper, E. De Poorter, I. Moerman, J. Verhaevert, P. Van Torre, and H. Rogier, "Bluetooth-low-energy-based fall detection and warning system for elderly people in nursing homes," *Journal of Sensors*, vol. 2022, no. 1, p. 9930681, 2022.

[14] X. Yu, B. Koo, J. Jang, Y. Kim, and S. Xiong, "A comprehensive comparison of accuracy and practicality of different types of algorithms for pre-impact fall detection using both young and old adults," *Measurement*, vol. 201, p. 111785, 2022.

[15] S. Lee, B. Koo, S. Yang, J. Kim, Y. Nam, and Y. Kim, "Fall-from-height detection using deep learning based on imu sensor data for accident prevention at construction sites," *Sensors*, vol. 22, no. 16, 2022.

[16] T. T. Um, F. M. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," in *Proceedings of the 19th ACM international conference on multimodal interaction*, 2017, pp. 216–220.

[17] K. M. Rashid and J. Louis, "Window-warping: a time series data augmentation of imu data for construction equipment activity identification," in *ISARC. Proceedings of the international symposium on automation and robotics in construction*, vol. 36. IAARC Publications, 2019, pp. 651–657.