Computers, Materials & Continua
DOI:10.32604/cmc.2022.022610
Article

Tech Science Press

TinyML-Based Fall Detection for Connected Personal Mobility Vehicles

Ramon Sanchez-Iborra[1], Luis Bernal-Escobedo[2], Jose Santa[3,*] and Antonio Skarmeta[2]

[1]University Center of Defense, General Air Force Academy, San Javier, 30720, Spain
[2]University of Murcia, Murcia, 30100, Spain
[3]Technical University of Cartagena, Cartagena, 30202, Spain
*Corresponding Author: Jose Santa. Email: jose.santa@upct.es
Received: 12 August 2021; Accepted: 22 September 2021

Abstract: A new wave of electric vehicles for personal mobility is currently crowding public spaces. They offer a sustainable and efficient way of getting around in urban environments, however, these devices bring additional safety issues, including serious accidents for riders. Thereby, taking advantage of a connected personal mobility vehicle, we present a novel on-device Machine Learning (ML)-based fall detection system that analyzes data captured from a range of sensors integrated on an on-board unit (OBU) prototype. Given the typical processing limitations of these elements, we exploit the potential of the TinyML paradigm, which enables embedding powerful ML algorithms in constrained units. We have generated and publicly released a large dataset, including real riding measurements and realistically simulated falling events, which has been employed to produce different TinyML models. The attained results show the good operation of the system to detect falls efficiently using embedded OBUs. The considered algorithms have been successfully tested on mass-market low-power units, implying reduced energy consumption, flash footprints and running times, enabling new possibilities for this kind of vehicles.

Keywords: Personal mobility; machine learning; on-board unit; C-ITS; IoT

1 Introduction

The arrival of eco-efficient mobility solutions is a reality in both dense urban areas and city outskirts [1]. Electric devices such as mopeds, segways, or scooters are currently massively employed for both transportation and leisure activities (Fig. 1). Nevertheless, the integration of these vehicles within the Cooperative-Intelligent Transportation Systems (C-ITS) ecosystem is far from being accomplished. This would enable the development of novel services and applications devoted to this specific vehicle segment, e.g., device monitoring, route planning, or traffic safety, among others. Regarding the latter, the number of accidents in which personal mobility devices are involved is notably increasing every year [2]. Key statements such as per 100,000 e-scooter trips, 20 people will end up with some sort of injury [3], evidence the importance of this new type of traffic accidents. This may be highly dangerous for riders, as they can suffer severe falls [4]. Recent studies reveal that these vehicles are proner to falls than regular bicycles, due to body dynamics and wheel size [5]. According to a recent study in [6], the type of injuries mainly involves the head (52%), arms (40%) and legs (20%), which indicate that these accidents can involve critical consequences. In fact, 22.5% of the injures types were identified as fractures. Currently, these vehicles are not provided with devices detecting any kind of accident and their connection with an external system is limited to a tracking system when they are rented. In this line, from the authors' perspective, automatic fall detection (AFD) systems may be critically useful for noticing these events and triggering an alarm to the corresponding emergency service using wireless communications.

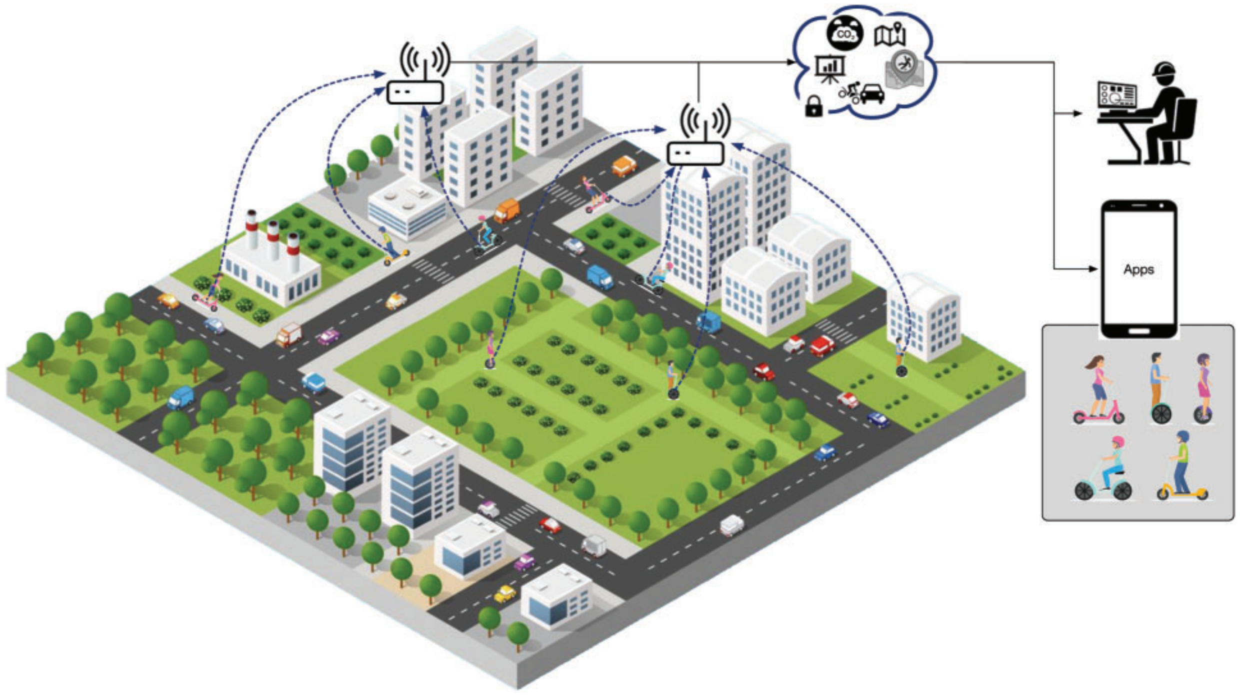

Figure 1: Personal mobility devices ecosystem in urban environments supported by mobile computing devices

On-board units (OBUs) are processing elements that are usually attached to vehicles aiming at enriching them with computing capabilities. This permits to run applications for supporting driving tasks and enhancing user experience [7]. OBUs embedded in traditional vehicles (i.e., cars or vans) do not usually present neither computation nor energy constraints, as they are powered by the vehicle battery. However, OBUs attached to personal mobility vehicles suffer from energy limitations, since they can have a built-in battery or be connected to the limited vehicle's power source. For that reason, both processing and memory capacities should be carefully saved in this case. This is the reason why hardware embedded in these units and application/system software should be designed to maintain a low operation profile. This is an issue for artificial intelligent algorithms needed by smart AFD systems, which are based on complex Machine Learning (ML) techniques requiring intensive data processing.

At this point is where the TinyML approach emerges [8]. This novel paradigm proposes to adapt ML models generated by widely-adopted frameworks such as TensorFlow or Scikit-learn, aiming at making them operational on constrained end-devices. Several alternatives are already publicly available in the market, e.g., Google's TensorFlow Lite or Microsoft's Embedded Learning Library (ELL), among many others. In this work, we propose an AFD mechanism to be embedded in an energy-constrained OBU. To this end, firstly, a large dataset has been generated, which includes data collected from real e-scooter rides, including provoked fall events. This dataset has been employed for training a set of ML algorithms, which have been integrated within an embedded OBU prototype following the TinyML paradigm for enabling on-device inference. Once a fall event has been detected, an early alert may be triggered by means of reliable and long-range communication interfaces such as those making use of Low Power Wide Area Network (LPWAN) technologies [9]. Therefore, the main contributions of this paper are the following:

- A public dataset for automatic detection of fall events in personal mobility devices.
- A discussion focused on the TinyML paradigm and its application to the sustainable mobility ecosystem.
- An ML-based fall detection system for personal mobility devices.
- A detailed study in terms of accuracy and footprint of a series of TinyML mechanisms integrated within a real OBU.

The contents of the paper are distributed as follows. Section 2 frames the work in the literature. Section 3 discusses the applicability and benefits of TinyML for providing smart services in the field of personal mobility vehicles. Section 4 describes the reference OBU used to gather data to generate ML models and later perform the evaluation. The TinyML-based fall detection system is explained in Section 5. The attained results are presented and discussed in Section 6. Finally, the paper is concluded in Section 7, summarizing the main findings and indicating future directions.

2 Background

The application of optimized ML-based solutions within the scope of eco-efficient mobility has been scarcely studied in the related literature. Given that we present a real implementation of an on-device fall detection system, in the following we explore the fields of human activity and fall detection and the use of embedded ML mechanisms for classification tasks.

2.1 Detection of Human Activity and Falls

Human Activity Recognition (HAR) and fall detection have been hot research topics since the advent of the IoT together with the integration of different types of low-cost sensors within smart-phones and other everyday gadgets. Work in [10], focused on elderly people monitoring, proposed using floor vibrations and sound to detect falls. The presented system uses signal processing and a pattern recognition algorithm to discriminate between falls and other events, gathering training data by mimicking falls using human dolls. In turn, more recent works made use of smart-phone's embedded sensors. For example, authors of [11] employed a simple algorithm employing acceleration measurements for detecting falls as well as long lie periods in order to reduce false positives. Work in [12] presented a more sophisticated detection method based on different ML models employing accelerometer data as well, hence demonstrating the usefulness of these low-cost and widespread smart-phone sensors for HAR and fall detection applications. Another approach using two independent devices for fall detection can be found in [13]. In this case, authors made use of a smart-phone and smart-watch for simultaneously tracking the status of the user by means of their embedded accelerometers and gyroscopes. The results showed a clear decrease on false positives while maintaining the effectiveness of the detection decisions. Other works in this field focused on video and image processing can be found in the survey work presented in [14].

In [15], authors demonstrated that model personalization, for example by using similarity between people related to physical aspects, notably improves the accuracy of HAR systems. Thus, a thorough study of the available data and user features is crucial for obtaining satisfactory results. Hence, the dataset is a key piece aiming at recognizing different patterns and situations. In the use-case under study in this work, obtaining a realistic dataset may be dangerous or hardly reproducible under laboratory conditions. Although different public and comprehensive HAR datasets can be found, e.g., [16], they focus on daily-live movement traces and not on riding activities. To tackle this issue, we have generated our own dataset collected from real rides (including falls) in order to produce different optimized and well-adjusted models for detecting falls in the specific use-case of electric personal mobility. As explained later, this dataset is publicly available for contributing to the HAR, ITS, and IoT research communities.

2.2 Embedded ML

ML is an area of high importance nowadays for the IoT ecosystem, given that many consumer applications can benefit from built-in advanced computing capabilities as they permit processing sensor data directly on the end-device. This leads to energy saving in communications and a reduction in application latency [17]. The integration of ML algorithms within IoT constrained elements implies extending the edge computing concept to the end-device/user itself, leading to the fog and mist computing paradigms [18]. However, ML-based mechanisms usually imply intensive computing tasks and, so far, they have required their execution on high-performance computers or clusters, sometimes aided by high-throughput GPU computing or application-specific integrated circuits (ASICs). As detailed in the following section, the TinyML paradigm permits to embed intelligence (ML mechanisms) in resource-constrained end-devices, hence, opening a plethora of opportunities for developing truly intelligent IoT units. Given the novelty of this concept [8], there is not any related literature addressing the application of TinyML models to the HAR or fall detection problems. However, some efforts proposing ML-based solutions to tackle these issues have been done, as these mechanisms have shown better performance than simpler threshold-based algorithms [19].

Work in [20] evaluated a set of ML classification algorithms, namely, k-Nearest Neighbors (k-NN, Naive Bayes, Artificial Neural Network (ANN), Support Vector Machine (SVM), and Least Squares Method (LSM), for the task of fall detection. Good accuracy over 85% was attained when distinguishing between daily activities and fall events, but authors did not present any results in terms of computing performance or memory footprint. Another approach can be found in [21], where an accident detection and classification system was developed by using data gathered from smart-phone's built-in sensors. Here a set of ML algorithms were implemented and evaluated within the smart-phone with good accuracy results, leveraging the superior processing power of these devices in comparison with common IoT units. Work in [22] presented a fuzzy inference system that, by gathering data collected from multiple sensors in an IoT environment, is able to infer that a fall has occurred. Authors emphasized the reliability of the solution given that multiple information sources are employed. Given the great range of available options to tackle the fall detection problem with ML-based solutions, works in [23,24] deeply explored the performance of different algorithms' and models' configurations in terms of accuracy but obviating a thorough evaluation of the computation and memory impact of the considered models.

Although additional works addressing ML-based automated fall and accident detection can be found in the literature [25], none of them permits to obtain an optimized mechanism that could be executed by an independent resource-constrained processing device. This is a critical aspect for our studied use-case, as the intelligent unit should be an embeddable solution for light and small vehicles such as electric scooters or segways not requiring external computation support. Thus, we propose to adopt the TinyML paradigm to build an intelligent and independent unit, capable of collecting the necessary data as well as conducting on-device ML inference for fall event detection. Besides, we also provide a comprehensive comparison of different TinyML models' footprint, performance and accuracy, extracted from a real proof of concept framed within the personal urban mobility ecosystem.

3 Smart Personal Mobility Vehicles and Machine Learning

The adoption of electric mobility solutions in our daily lives is a reality thanks to the great range of benefits brought by these devices. On the one hand, eco-efficient vehicles can replace traditional transportation means in cities due to their reduced carbon footprint as well as their mobility flexibility in crowded scenarios. Recent access restrictions to the city center in Europe, such as the cases of London, Paris or Madrid, for regular combustion vehicles, lead to the adoption of sustainable vehicles that complement regular bicycles. In addition, devices such as electric scooters and bikes, segways, etc. are also being employed for outdoor recreational activities. As it has been done for regular vehicles, the attachment of OBUs to personal mobility vehicles will allow their integration within the C-ITS ecosystem. This will boost the development of novel services for this vehicle segment. Besides, embedding intelligence in these processing units will make them evolve from simply connected elements to truly smart "things", hence improving the eco-mobility experience and riders safety. To this end, the TinyML paradigm will fill the gap that currently hinders the implementation of ML-based mechanisms within these constrained computing platforms.

3.1 TinyML

The recent emergence of the TinyML paradigm [8] has enabled the integration of powerful ML schemes within constrained processing devices such as those powered by microcontrollers [26]. Although previous advances have been achieved in this field, the attained solutions are usually ad-hoc implementations addressing specific use-cases for a given processing board, which clearly limits the generalization capability of this approach. As depicted in Fig. 2, TinyML allows to convert ML models generated by well-known toolkits, e.g., TensorFlow, Keras, PyTorch, Scikit-learn, etc., in order to make them executable in constrained platforms. Once the pre-trained model is deployed on the resource-limited unit, the ML algorithm can be executed to cover the target use case.

From a technical perspective, embedding intelligence in end-devices will bring interesting features to IoT deployments. Firstly, current constrained devices notably depend on their connection with the cloud in order to send collected data and/or receive commands. As communication activities can be more energy demanding that moderate processing tasks in these nodes, performing local computation for making smart decisions can be more adequate than relying on remote processing. This aspect is not negligible given that a great part of IoT devices is powered by batteries. Besides, reducing raw data transmissions further diminish cybersecurity risks and reduce computing impact of cryptography operations with their associated energy consumption.

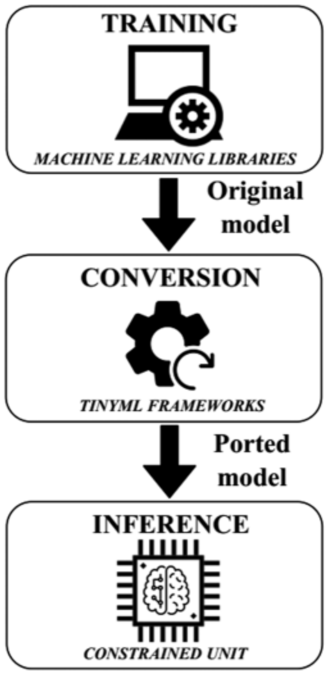

Figure 2: Machine learning conversion workflow to pass from a general-purpose model to a TinyML model for constrained devices

Some TinyML frameworks are already publicly available in the market. The interest of technological companies in this paradigm is clear, as Google, Microsoft, or ARM have released different toolkits. Google's TensorFlow Lite and Microsoft's ELL permit to transform Deep Learning (DL) models from TensorFlow and the Microsoft Cognitive Toolkit (CNTK), respectively, to C++ code that can be compiled for 32-bit processors such as the ARM Cortex-M series. Devoted to this processor as well, ARM has released the Cortex Microcontroller Software Interface Standard-NN (CMSIS-NN) framework, which is interoperable with models from TensorFlow and Caffe. Besides, ARM has also developed the ARM-NN library, which widens the range of supported boards towards ML-oriented Ethos neural processing units such as ARM Mali GPUs and ARM Cortex-A CPUs.

As shown in Tab. 1, others interesting TinyML frameworks are also available for enabling interoperability with additional ML libraries and constrained boards. For example, emlearn[1] permits to port a set of ML models, namely, Multi-Layer Perceptron (MLP), Decision Tree (DT), Random Forest (RF), or Gaussian Naive Bayes (GNB) generated by Scikit-learn or Keras, for making them runnable in 8-bit platforms such as the widely-adopted Arduino Uno. Also compatible with this development board, the MicroMLGen framework ports Support Vector Machine (SVM) models produced by Scikit-learn. As a result, a variety of models generated by robust ML libraries are now at our disposal to be integrated within constrained processing units. This will foster the development of a vibrant TinyML community, which will grow until forming a rich and productive ecosystem.

In the light of the previous discussion, the TinyML paradigm, driven by the available frameworks, will permit to turn current end-devices on smart elements, therefore evolving the traditional IoT landscape to the novel Internet of Intelligent Things (IoIT). Thanks to this revolution, a new wave

[1] https://github.com/emlearn/emlearn

of applications will be designed and implemented. In the following we explore the range of potential services to be developed in the area of sustainable personal mobility.

Table 1: TinyML frameworks reviewed

Framework	Developer	Compatible ML libraries	Output languages	Target platforms
TensorFlow Lite[2]	Google	TensorFlow	C++ 11	ARM Cortex-M
ELL[3]	Microsoft	CNTK, Darknet, ONNX	C/C++	ARM Cortex-M/A, Arduino, micro:bit
CMSIS-NN[4]	ARM	TensorFlow, Caffe, PyTorch	C99	ARM Cortex-M
ARM-NN[5]	ARM	TensorFlow, Caffe, ONNX	C	ARM Cortex-A, ARM Mali, ARM Ethos
Emlearn[6]	Particular developer	Keras, Scikit-learn	C	AVR Atmega, ESP8266, Linux
MicroMLGen[7]	Particular developer	Scikit-learn	C	Arduino, ESP8266, ESP32

3.2 *Enabled Services*

Traffic safety will be one of the most benefited fields of application with the integration of intelligent schemes in constrained OBUs. One of the most critical services within this scope is automatic accident detection, which is deeply investigated in the following sections. A quick detection of falls is crucial for a fast reaction in order to notify the event and the accident location to the closest emergency service. Although some efforts have been devoted to the development of AFD systems for pedestrians [27] and bike riders [28], the new range of electric personal mobility vehicles has not been investigated so far, as studies such as the one in [29] reveal. Besides this application, the intelligent interconnection of personal mobility devices to the C-ITS ecosystem will boost the development of vehicle-to-vehicle (V2 V) collision avoidance systems such as those proposed for two-wheelers [30], and the implementation of advanced interaction schemes of personal vehicles with the traffic infrastructure [31].

Another area of application is on-board intelligent vehicle diagnosis [32]. By introducing ML mechanisms that could inspect different vehicle parameters, it will be able to perform self-inspections for prompt fault detection. This feature will permit to prevent mechanical failures and auto-adjust the device in case of software-solvable issues. In this line, intelligent device status tracking will be also enabled; hence, vehicle telemetry will be accessible for analysis from users' devices such as smart-phones or tablets. Offline route planning is another interesting service for providing riders with tailored routes according to learned habits, cached digital maps, and special characteristics of personal mobility

[2] https://www.tensorflow.org/lite/
[3] https://microsoft.github.io/ELL/
[4] https://arm-software.github.io/CMSIS_5/General/html/
[5] https://github.com/ARM-software/armnn
[6] https://github.com/emlearn/emlearn
[7] https://github.com/eloquentarduino/micromlgen

vehicles. Thus, this application could indicate, for example, green tracks or alternative safer routes that avoid dense traffic situations. Once parked, intelligent services related to device security will be powered by ML mechanisms. A great number of these vehicles are stolen every day, therefore the implementation of intelligent e-locks relaying on user's biometric data will be of great interest. Traditional mechanical locks will be replaced by smart systems that may block the device upon manipulation. Besides, real-time alerts can be triggered to the vehicle's owner if a steal-attempt is detected.

Advanced real-time scenario monitoring will be also useful for warning riders against unhealthy pollution conditions. Continuous city sensing will help smart city platforms to collect pre-processed data from streets. Hence, a plethora of recommendation services will be enabled for improving citizens well-being, at the same time that communication resources and battery are saved. Finally, aligned with the recreational perspective of these vehicles, smart infotainment applications will be also powered by the use of ML, e.g., music recommendations, audio guides and books, or contextual advice.

4 Reference on-Board Unit

In previous work carried out by authors, a functional and efficient OBU for personal mobility vehicles was developed [1]. This OBU has been considered in this work to initially get training data and later validate and test the proposed TinyML-based AFD system. Its main features are depicted in Fig. 3. It is a low-power consumption device provided with an off-the-shelf single board computer (SBC) equipped with Low-Power Wide-Area Network (LPWAN) capabilities. Concretely, Long Range Wide-Area Network (LoRaWAN) and Narrow-Band IoT (NB-IoT) technologies are used for the sake of minimizing energy consumption and obtaining long-range and reliable communication links. Using either LoRaWAN or NB-IoT, the OBU can access cloud services through the Internet, which can feed web applications or smart-phone apps, for instance, offering Cooperative, Connected and Automated Mobility (CCAM) advances.

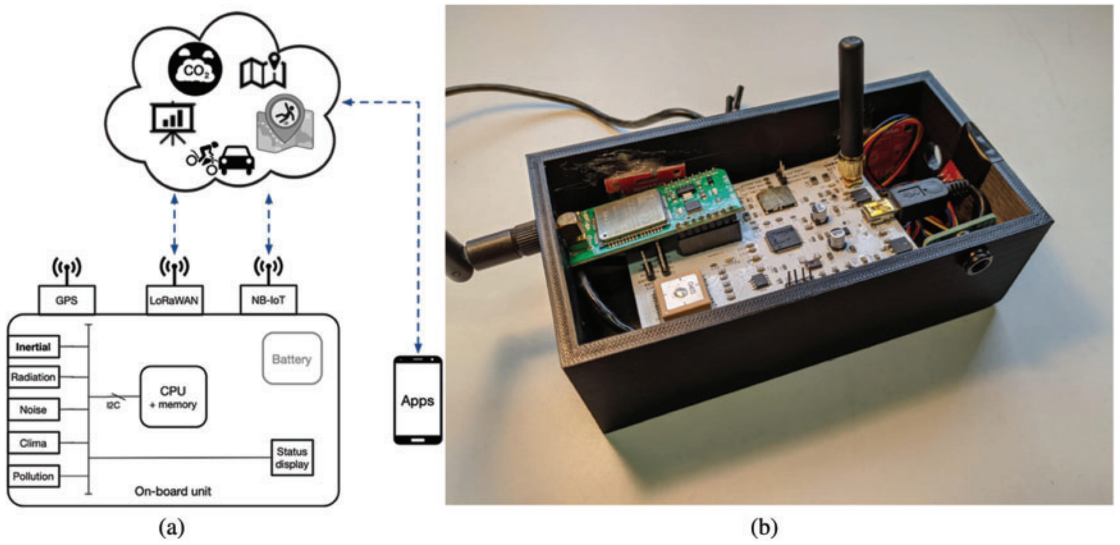

Figure 3: Architecture and prototype of the OBU used for e-scooters. (a) OBU architecture design with sensors, processing unit, basic I/O and communication transceivers (b) Functional prototype developed for validation and testing purposes

The prototype implemented (right side of Fig. 3) includes a Raspberry Pi Zero W SBC, with a System on Chip (SoC) powered by a 1 GHz ARM11 CPU coupled with 512 MB of SDRAM. The top white part is a communication board integrating a GPS receiver, a LoRaWAN transceiver (Murata CMWX1ZZABZ-093) and a NB-IoT modem (Quectel BC95). A 5 Ah lithium battery (RS PRO PB-A5200) is used to reach running times over 48 h reporting data continuously. The unit case has been manufactured using a 3D printer and its dimensions are 15 × 7 × 5 cm. Among the various sensors installed in the unit, the most important one in the studied use case is the 9 degrees of freedom (DOF) inertial unit ST LSM9DS1. It includes a three-axis accelerometer, a three-axis gyroscope and three-axis magnetometer. Hence, it provides angular velocity, acceleration and heading, which are the input of the developed AFD system.

Given the wide adoption of Raspberry solutions in embedded electronics and, particularly, when an agile prototyping is necessary in research tasks in Academia, we have opted for a Pi Zero platform. It provides good performances under low power consumption, good I/O capabilities and the market is plenty of sensory solutions for Raspberry. Additionally, it is able to execute an embedded operating system, which makes it convenient to deploy several applications and, in our case, different TinyML models. Although microcontroller-based designs could be developed to run services and further reduce energy consumption, their flexibility for software upgrading and multi-task purposes would be much more limited.

5 TinyML-Based Fall Detection

This section presents a specific TinyML demonstrator in which we address the AFD issue in personal mobility vehicles by developing different ML models that are adapted to constrained devices. In our particular evaluation, we consider the OBU described above to perform an experimental evaluation of the solution.

5.1 Dataset

The first step to develop the proposed AFD system is creating a rich and large dataset for training and testing the generated models. To this end, a sampling campaign has been carried out with the reference OBU recording about three hours of urban e-scooter riding and 50 provoked falls in a controlled environment, including left and right overturns and front collisions at moderate speed. Fig. 4 includes real photos taken during the trials, for the cases of left falls and forward collision. It is indicated the initial position of the scooter, the direction of the movement and its final status after the trial. For the forward collision trials, the scooter was manually propelled at 10 km/h to collide with a mat. The scooter used was an Outsider E-Volution 8,5 Phoenix, from Cecotec. The user was the same along the regular sampling process, covering different road types and speed/turning conditions. All trials have been carried out in the city center of Cartagena (Spain), driving on regular roads and cycle lanes.

Data were recorded from the inertial sensor unit integrated in the OBU in order to obtain three-axis acceleration (a_{xi},a_{yi},a_{zi}) and angular velocity (θ_{xi},θ_{yi},θ_{zi}) at a sampling rate of 10 Hz. Considering these features, the obtained raw data vector presents six elements. We have considered the three axis components given that falls in personal mobility vehicles may happen in any direction [5]. After the sampling campaign, the obtained dataset was examined and missing or corrupted samples were fixed.

For the sake of experiment repeatability and reproducibility as well as for enabling further related studies, the resulting dataset has been publicly released[8].

Besides, aiming at evaluating the performance of the considered models with different input feature vectors, we also compute the corresponding signal magnitude vector (SMV) of both monitored parameters as indicated in Eqs. (1) and (2).

$$a_i = \sqrt{a_{xi}^2 + a_{yi}^2 + a_{zi}^2} \tag{1}$$

$$\theta_{tilt\,i} = \sqrt{\theta_{xi}^2 + \theta_{yi}^2 + \theta_{zi}^2} \tag{2}$$

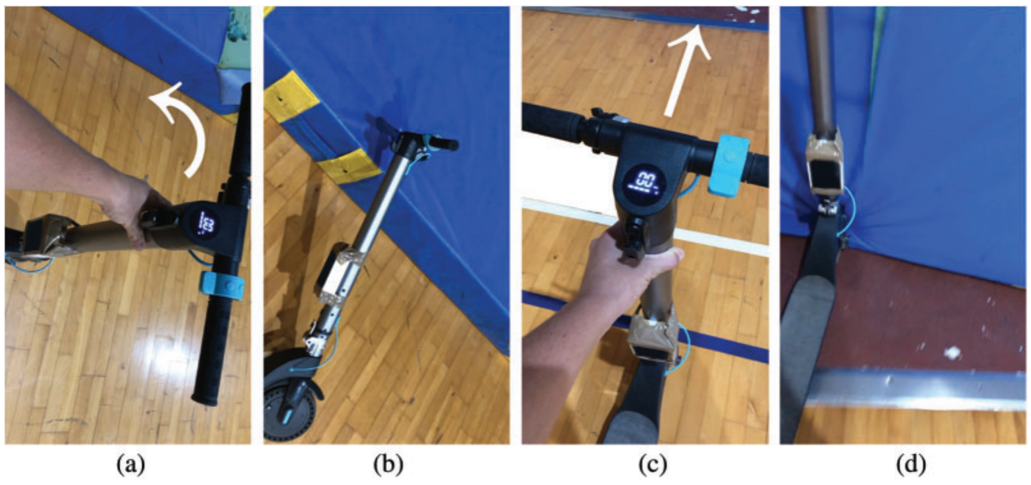

Figure 4: Procedure followed to obtain dynamics data from falls with the e-scooter. (a) Fall to left, initial status (b) Fall to left, final status (c) Frontal collision, initial status (d) Frontal collision, final status

Given that captured measurements may suffer isolated sudden changes due to shocks caused by speed bumps or road imperfections, we finally aggregate every second, i.e., 10 samples, the captured raw data and the corresponding computed SMVs and calculate their variance. Thus, the final feature vectors that are passed to the ML algorithms are the following: ($\sigma 2(axi)$, $\sigma 2(ayi)$, $\sigma 2(azi)$, $\sigma 2(\theta xi)$, $\sigma 2(\theta yi)$, $\sigma 2(\theta zi)$) or ($\sigma^2(a_i)$, $\sigma^2(\theta_{tilt\,i})$), depending on the selected size of the input vector, namely, six or two elements, respectively. With these input vectors, the decision algorithm outputs if a fall detected or not.

5.2 TinyML Models

In order to evaluate the adequacy of different ML algorithms for the AFD task, we have followed the workflow of Fig. 2 previously described. We have made use of the well-known Scikit-learn toolkit[9] to train and generate a variety of ML models, namely, Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), Decision Tree (DT), Gaussian Naive Bayes (GNB), and Random Forest (RF). Given the processing and memory constraints of the OBU presented above, these models have been

[8] https://gitlab.com/scooter-project/ml-dataset
[9] https://scikit-learn.org

optimized by using different TinyML frameworks. Concretely, MicroMLGen has been used to adapt the SVM models, while the versatile emlearn library has been employed to port the rest of models generated by Scikit-learn. The dataset generated as described above has been split using stratification into three subsets: 70% for training and 20% for validation in Scikit-learn, and 10% for testing in the OBU. As performance metrics, we have chosen the accuracy level of the generated models, their memory footprint, the required processing time and energy consumed once deployed in the OBU.

5.3 Working Flow

The overall operation of the system using any of the adopted TinyML models is depicted in the flow chart included in Fig. 5. After beginning the ride, the software starts collecting data from inertial sensors continuously at a sampling rate of 10 Hz, as aforementioned. Once the samples collected fill the reference window size (10 samples in our experiments), data are aggregated to be processed by the TinyML model. This module is in charge of detecting a fall. If this is not the case, a new data collection cycle is started; whereas if a fall is detected, an alarm message is transmitted using the communication module of the OBU towards a given alert management system in the cloud. Under normal operation with no falls, the system would be running and continuously gathering data and executing the TinyML-based algorithm in a continuous cycle.

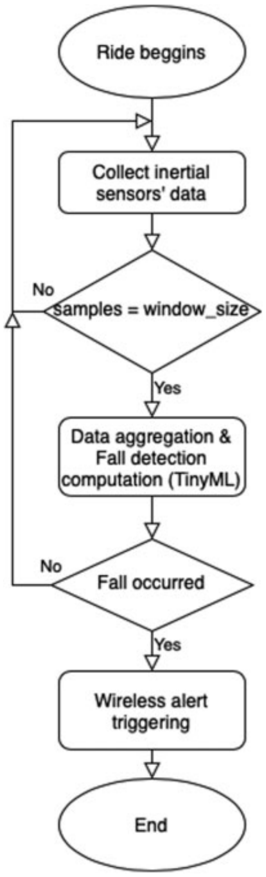
Ride beggins
Collect inertial sensors' data
No
samples = window_size
Yes
Data aggregation & Fall detection computation (TinyML)
No
Fall occurred
Yes
Wireless alert triggering
End

Figure 5: Flow chart of the operation of the TinyML-based fall-detection solution

6 Results

6.1 Overall Comparison of TinyML Models

The obtained results in terms of the identified figures of merit are shown in Tab. 2. Different configurations have been tested for each of the algorithms although we only present the most adequate ones for the sake of clarity. As can be seen in the last column, the accuracy achieved by all the considered algorithms is similar and their performance is notably successful (around 0,96). For most of the cases, it can be seen that the computing time is noticeably low, in the order of tens of microseconds, except for the case of the SVM, which is the slowest algorithm. This is explained by its greater complexity in comparison with the other algorithms, also evidenced by the size of the generated binary file due to the kernels are defined as float random variables. SVM also provides the lowest accuracy levels when using the Sigmoid kernel, so its usage is not recommended.

It is noticeable the high accuracy offered by RF (0,97) when using five estimators and six input features, and requiring an execution time of only 11 ± 14 μs. Regarding MLP models, the different evaluated network configurations provide similar accuracy levels, although the processing time and the required RAM increase with the number of neurons forming the MLP architecture because, differently from the SVM implementation, these values are stored in volatile memory. Finally, GNB presents greater accuracy than DT but longer processing times with both two or six input features. As mentioned above, the accuracy detecting falls of all of the evaluated algorithms is remarkable.

Aiming at providing insights regarding the validity of the results, the F1-score attained for each model is also included in Tab. 2. Observe that the obtained F1 scores are aligned with the accuracy values, hence indicating the good performance of the models avoiding false positives or negatives. This is of prominent importance in the application under study given that a non-detected fall may lead to serious issues for the injured driver.

Table 2: Execution time, on-device footprint, accuracy, and F1-score of the evaluated TinyML models

Algorithm	Model characteristics	Input features	Time (μs)	Power (nAh)	RAM (kB)	Executable size (kB)	Accuracy	F1
Decision tree (DT) max. depth = 12	Criterion = entropy	2	6 ± 2	0,49	1404	15,50	0,9505	0,9463
		6	7 ± 2	0,57	1452	14,79	0,9653	0,9631
	Criterion = gini	2	6 ± 2	0,49	1364	15,53	0,9493	0,9444
		6	7 ± 2	0,57	1456	14,75	0,9621	0,9611
Gaussian naive bayes (GNB)		2	8 ± 2	0,65	1404	9,54	0,9621	0,9539
		6	14 ± 2	1,14	1456	9,63	0,9634	0,9587
Random forest (RF) criterion = gini max. depth = none	2 estimators	2	7 ± 2	0,57	1420	30,77	0,9332	0,9384
		6	8 ± 2	0,65	1368	20,66	0,9531	0,956
	5 estimators	2	9 ± 2	0,73	1416	66,13	0,9608	0,9545
		6	11 ± 3	0,89	1436	45,83	0,9717	0,9697
Support vector machine (SVM) gamma = 0.0001	Linear kernel	2	175 ± 4	14,29	1468	57,51	0,9602	0,9456
		6	378 ± 6	30,88	1576	77,51	0,9672	0,9592

(Continued)

Table 2: Continued

Algorithm	Model characteristics	Input features	Time (μs)	Power (nAh)	RAM (kB)	Executable size (kB)	Accuracy	F1
	Sigmoid kernel	2	293 ± 8	23,94	1660	57,51	0,9261	0,9231
		6	553 ± 9	45,18	1632	93,51	0,9268	0,9252
Multi-layer perceptron (MLP) activation function = tanh max. iterations = 4000	Hidden layers = 1 Neurons per layer = 2	2	12 ± 5	0,98	1492	9,53	0,9608	0,9512
		6	14 ± 5	1,14	1524	9,53	0,9608	0,9330
	Hidden layers = 1 Neurons per layer = 5	2	14 ± 4	1,14	1500	9,53	0,9550	0,9330
		6	18 ± 5	1,47	1516	9,53	0,9679	0,9616
	Hidden layers = 1	2	18 ± 5	1,47	1492	9,53	0,9615	0,9517
	Neurons per layer = 10	6	24 ± 6	1,96	1524	9,53	0,9634	0,9570
	Hidden layers = 2 Neurons per layer = 2	2	14 ± 5	1,14	1492	9,53	0,9550	0,9331
		6	16 ± 5	1,30	1552	9,53	0,9615	0,9572
	Hidden layers = 2 Neurons per layer = 5	2	19 ± 5	1,55	1524	9,53	0,9608	0,9518
		6	23 ± 5	1,87	1520	9,53	0,9666	0,9606
	Hidden layers = 2 Neurons per layer = 10	2	31 ± 5	2,53	1520	9,53	0,9595	0,9496
		6	38 ± 5	3,10	1528	9,53	0,9653	0,9603

Regarding the footprint of the implementations on the target processing board, i.e., Raspberry Pi Zero in our case, it can be seen the low impact on both RAM and flash memories, hence making feasible to run all the tested ML classifiers with real time sensor data on such a constrained OBU. The moderate RAM required permits the generated models to coexist with other simultaneous functions running in the unit. Besides, the limited size of the binary files built for the target platform makes them suitable to be embedded in the considered unit as well as other more resource-limited boards, e.g., microcontroller-based architectures, without the risk of consuming a significant part of the available flash memory.

Given the relevance of power consumption in the target environment, indicative values are also given in Tab. 2. They have been obtained taking as reference the consumption of the unit under 100% CPU usage together with sensor reads at a rate of 1 Hz. The value obtained after a set of consecutive execution of ML algorithms was 294,17 ± 14,49 mA, not observing differences when executing the different models. Hence, considering the execution time of each model, it has been computed the power consumption (in nanoamperes per hour-nAh). Results in Tab. 2 showcase the limited energy needed by most of the cases except for SVM, due to the aforementioned algorithm complexity.

Considering the discussion above, GNB, DT, or RF algorithms would be the most recommended options to implement the embedded fall detection mechanism, as they present very high accuracy with a reduced impact in terms of processing latency, memory requirements, and power consumption. Additionally, SVM and MLP produce slower and heavier models as compared to GNB, DT, and RF, without providing a better performance in terms of accuracy.

6.2 *Impact of Number of Input Features*

Fig. 6 depicts the models' footprint and performance variation depending on the number of inputs (two or six). Concretely, we show the relative difference of the evaluated metrics when the classifiers receive feature vectors of two or six elements, taking the former as the baseline. We find that models with six inputs present an almost negligible accuracy improvement, however, this is achieved at the expense of increasing the algorithms' running time. It can be observed that in every case, models with six input features require much more processing time than their analogous models with two inputs. This fact is especially remarkable for GNB, MLP, and SVM, with more than 20% increase in the algorithm's computation time.

Flash memory usage especially increases with six input features in SVM. This confirms the previous discussion regarding memory requirements of support vectors. Models based on GNB and MLP do not suffer a noticeable variation in their memory footprint depending on the number of input features. In fact, all MLP models have an identical executable file size and similar RAM footprints. This is due to the binary file padding and how the code works in this implementation (by iterating through a list of neurons), so the binary files only differ in the quantity of biases and weights that describe the network, which are really small, e.g., less than one kB for a $10 + 10 + 5$ neurons model and less than 100 bytes for the two neurons model. These small differences are further reduced by the executable file padding, used to align different memory blocks in accordance with the target computer architecture. Additionally, optimizing memory accesses and code makes RAM usage to be almost the same on the small networks we have used.

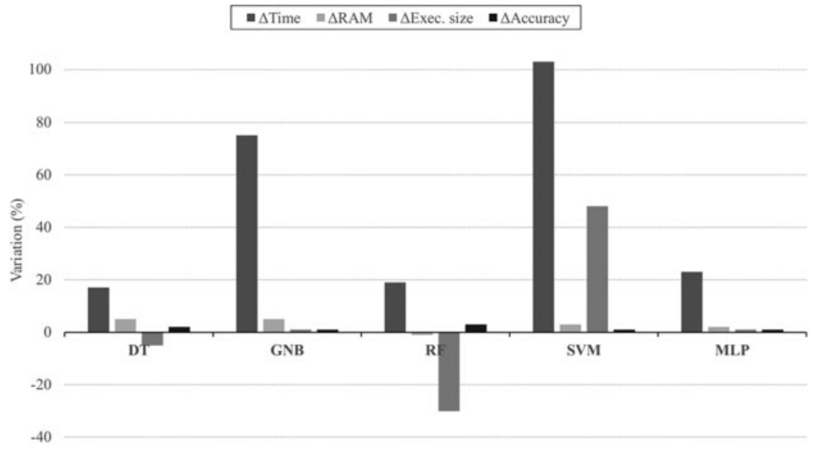

Figure 6: Models' footprint and performance variation according to the number of input features (6 *vs.* 2), taking as a reference the results obtained with two inputs

The cases of DT and RF are worth further analysis. In both of them a decrease in the required flash memory is obtained when increasing the number of inputs, especially in the case of RF. This behavior can be explained by the fact that the six inputs models are able to find a solution with lower tree or forest complexity in comparison with models receiving feature vectors of two elements. Regarding the impact on RAM, a similar behavior is attained for RF, while a small increase in volatile memory demands is detected for DT. In the light on these and previous outcomes, together with the high accuracy achieved, we recommend the use of RF with five estimators to build a precise and efficient ML-based AFD system.

Overall, from the obtained results it can be extracted that the TinyML paradigm opens a plethora of opportunities for integrating ML-based mechanisms on end-devices with resource limitations. Besides, the low footprint detected enables interesting features such as over the air (OTA) updates of ML models, thus offloading the heavy task of training and upgrading the ML system onto a cloud or Multi-Access Edge Computing (MEC) node; and even further distribute training tasks by exploiting federated learning capabilities while reducing communication requirements.

7 Conclusion

The traditional traffic landscape consisting of fuel-powered vehicles is being enriched with a plethora of eco-efficient devices that provide users with more sustainable mobility options. However, given the novelty of personal mobility vehicles such as e-scooters, segways, or electric wheels, among others, there are still different aspects related to their tracking, connectivity, and safety that should be promptly addressed. In this work, we have explored the improvement of driver safety under falls or accidents in this segment by proposing a fall detection system powered by ML. Concretely, the state-of-the-art TinyML paradigm is exploited to develop an ML-based efficient and accurate fall detection system. To this end, a riding/fall dataset has been produced (and publicly released) to build a range of TinyML models considering different algorithms, namely, RF, DT, SVM, GNB and MLP, considering a range of configurations. These data have been obtained from real trials with an electric scooter in the city of Cartagena (Spain), and they have been used for training, validation and tests. Hence, these models have been deployed on a lightweight and connected OBU prototype in order to analyze the attained accuracy when detecting a fall, as well as evaluating the models' footprint on the host processing unit. The results validate the good operation of the proposal, as the attained classifiers' performance permits to quickly and accurately detect fall events, while the limited memory requirements and energy consumption of the produced models ensure their integration in a plethora of constrained processing units. As future work, we plan to generate the TinyML models for even more limited computing boards as well as explore innovative ML paradigms such as federated learning to speed-up algorithm generation and sharing. Besides, further analysis on the sensed data may permit to extract more sophisticated outcomes such as accident type or severity.

Funding Statement: This work has been supported by the Spanish Ministry of Science, Innovation and Universities, under the Ramon y Cajal Program (ref. RYC-2017-23823) and the projects ONOFRE 3 (ref. PID2020-112675RB) and Go2Edge (ref. RED2018-102585-T); by the European Commission, under the 5G-MOBIX (ref. 825496) project; and by the Spanish Ministry for the Ecological Transition and the Demographic Challenge, under the MECANO project (ref. PGE-MOVES-SING-2019-000104). Ramon Sanchez-Iborra would like to thank Prof. Sanchez-Muñoz for his patience and valuable comments during the preparation of this paper.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Santa, L. Bernal-Escobedo and R. Sanchez-Iborra, "On-board unit to connect personal mobility vehicles to the IoT," in *17th Int. Conf. on Mobile Systems and Pervasive Computing (MobiSPC)*, vol. 175, pp. 173–180, 2020.

[2] Radiological Society of North America (RSNA), "New study looks at motorized scooter injuries," RSNA Press Release, 2019.

[3] Austin Public Health (APH), "Dockless electric scooter-related injuries study," APH Epidemiology and Disease Surveillance Unit, 2019.

[4] T. K. Trivedi, C. Liu, A. L. Antonio, N. Wheaton, V. Kreger *et al.,* "Injuries associated with standing electric scooter use," *JAMA Network Open*, vol. 2, no. 1, pp. e187381, 2019.

[5] M. Paudel, F. F. Yap and A. K. Bastola, "Safety assessment of personal mobility devices with different wheel size based on their dynamic stability performance," *International Journal of Sustainable Design*, vol. 3, no. 4, pp. 227, 2020.

[6] J. Y. Kim, S. C. Lee, S. Lee, C. A. Lee, K. O. Ahn *et al.,* "Characteristics of injuries according to types of personal mobility devices in a multicenter emergency department from 2011 to 2017," *Medicine*, vol. 100, no. 6, pp. e24642, 2021.

[7] M. A. Javed and E. Ben Hamida, "On the interrelation of security, QoS, and safety in cooperative ITS," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1943–1957, 2017.

[8] P. Warden and D. Situnayake, in *Tinyml: Machine Learning with Tensorflow Lite on Arduino and Ultra-Low-Power Microcontrollers*, 1st ed. Sebastopol, USA: O'Reilly Media, 2020.

[9] R. Sanchez-Iborra and M. -D. Cano, "State of the art in LP-wAN solutions for industrial IoT services," *Sensors*, vol. 16, no. 5, pp. 708, 2016.

[10] Y. Zigel, D. Litvak and I. Gannot, "A method for automatic fall detection of elderly people using floor vibrations and sound—proof of concept on human mimicking doll falls," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 12, pp. 2858–2867, 2009.

[11] T. Tri, H. Truong and T. Khanh, "Automatic fall detection using smartphone acceleration sensor," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 12, pp. 123–129, 2016.

[12] S. Wan, L. Qi, X. Xu, C. Tong and Z. Gu, "Deep learning models for real-time human activity recognition with smartphones," *Mobile Networks and Applications*, vol. 25, no. 2, pp. 743–755, 2020.

[13] E. Casilari and M. A. Oviedo-Jiménez, "Automatic fall detection system based on the combined use of a smartphone and a smartwatch," *PLOS ONE*, vol. 10, no. 11, pp. e0140929, 2015.

[14] L. Minh Dang, K. Min, H. Wang, M. Jalil Piran, C. Hee Lee *et al.,* "Sensor-based and vision-based human activity recognition: A comprehensive survey," *Pattern Recognition*, vol. 108, pp. 107561, 2020.

[15] A. Ferrari, D. Micucci, M. Mobilio and P. Napoletano, "On the personalization of classification models for human activity recognition," *IEEE Access*, vol. 8, pp. 32066–32079, 2020.

[16] E. Casilari, J. A. Santoyo-Ramón and J. M. Cano-García, "UMAFall: A multisensor dataset for the research on automatic fall detection," *Procedia Computer Science*, vol. 110, pp. 32–39, 2017.

[17] R. Sanchez-Iborra and A. F. Skarmeta, "Tinyml-enabled frugal smart objects: Challenges and opportunities," *IEEE Circuits and Systems Magazine*, vol. 20, no. 3, pp. 4–18, 2020.

[18] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. Goren *et al.,* "Fog computing conceptual model," *Gaithersburg*, MD, 2018.

[19] O. Aziz, M. Musngi, E. J. Park, G. Mori and S. N. Robinovitch, "A comparison of accuracy of fall detection algorithms (threshold-based vs. machine learning) using waist-mounted tri-axial accelerometer signals from a comprehensive set of falls and non-fall trials," *Medical & Biological Engineering & Computing*, vol. 55, no. 1, pp. 45–55, 2017.

[20] P. Vallabh, R. Malekian, N. Ye and D. C. Bogatinoska, "Fall detection using machine learning algorithms," in *24th Int. Conf. on Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Croatia, pp. 1–9, 2016.

[21] N. Kumar, D. Acharya and D. Lohani, "An IoT-based vehicle accident detection and classification system using sensor fusion," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 869–880, 2021.

[22] S. Moulik and S. Majumdar, "Fallsense: An automatic fall detection and alarm generation system in IoT-enabled environment," *IEEE Sensors Journal*, vol. 19, no. 19, pp. 8452–8459, 2019.

[23] A. Chelli and M. Patzold, "A machine learning approach for fall detection and daily living activity recognition," *IEEE Access*, vol. 7, pp. 38670–38687, 2019.

[24] I. Putra, J. Brusey, E. Gaura and R. Vesilo, "An event-triggered machine learning approach for accelerometer-based fall detection," *Sensors*, vol. 18, no. 2, pp. 20, 2017.

[25] A. Ramachandran and A. Karuppiah, “A survey on recent advances in wearable fall detection systems,” *BioMed Research International*, vol. 2020, pp. 1–17, 2020.
[26] C. Bormann, M. Ersue and A. Keranen, “Terminology for constrained-node networks,” IETF RFC 7228, 2014.
[27] L. Ren and Y. Peng, “Research of fall detection and fall prevention technologies: A systematic review,” *IEEE Access*, vol. 7, pp. 77702–77722, 2019.
[28] H. -R. Choi, M. -H. Ryu, Y. -S. Yang, N. -B. Lee and D. -J. Jang, “Evaluation of algorithm for the fall and fall direction detection during bike riding,” *International Journal of Control and Automation*, vol. 6, no. 6, pp. 209–218, 2013.
[29] F. Wirth, T. Wen, C. Fernandez-Lopez and C. Stiller, “Model-based prediction of two-wheelers,” in *IEEE Intelligent Vehicles Symposium (IV)*, Las Vegas, NV, USA, pp. 1669–1674, 2020.
[30] J. Santa and P. J. Fernández, “Seamless IPv6 connectivity for two-wheelers,” *Pervasive and Mobile Computing*, vol. 42, pp. 526–541, 2017.
[31] S. Chen, J. Hu, Y. Shi, Y. Peng, J. Fang *et al.,* “Vehicle-to-everything (v2x) services supported by LTE-based systems and 5G,” *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 70–76, 2017.
[32] M. Al-Zeyadi, J. Andreu-Perez, H. Hagras, C. Royce, D. Smith *et al.,* “Deep learning towards intelligent vehicle fault diagnosis,” in *Int. Joint Conf. on Neural Networks (IJCNN)*, Glasgow, UK, pp. 1–7, 2020.