applied sciences

MDPI

Article

Dataset and System Design for Orthopedic Walker Fall Detection and Activity Logging Using Motion Classification

Maxwell Huang [1] and Antony Garcia [2,3,*]

[1] Noble and Greenough School, Dedham, MA 02026, USA
[2] Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA 01609, USA
[3] Campus Víctor Levi Sasso, Universidad Tecnológica de Panamá, Panama City 0819-07289, Panama
* Correspondence: agarcia3@wpi.edu or antony.garcia@utp.ac.pa

Abstract: An accurate, economical, and reliable device for detecting falls in persons ambulating with the assistance of an orthopedic walker is crucially important for the elderly and patients with limited mobility. Existing wearable devices, such as wristbands, are not designed for walker users, and patients may not wear them at all times. This research proposes a novel idea of attaching an internet-of-things (IoT) device with an inertial measurement unit (IMU) sensor directly to an orthopedic walker to perform real-time fall detection and activity logging. A dataset is collected and labeled for walker users in four activities, including idle, motion, step, and fall. Classic machine learning algorithms are evaluated using the dataset by comparing their classification performance. Deep learning with a convolutional neural network (CNN) is also explored. Furthermore, the hardware prototype is designed by integrating a low-power microcontroller for onboard machine learning, an IMU sensor, a rechargeable battery, and Bluetooth wireless connectivity. The research results show the promise of improved safety and well-being of walker users.

Keywords: orthopedic walker; dataset; IoT; fall detection; activity logging; inertial measurement unit; machine learning; deep learning

Citation: Huang, M.; Garcia, A. Dataset and System Design for Orthopedic Walker Fall Detection and Activity Logging Using Motion Classification. *Appl. Sci.* **2023**, *13*, 11379. https://doi.org/10.3390/app132011379

Academic Editors: Mohamed Medhat Gaber and Mohammed Abdelsamea

Received: 21 September 2023
Revised: 12 October 2023
Accepted: 14 October 2023
Published: 17 October 2023

1. Introduction

According to a report from the World Health Organization (WHO), the percentage of the world's population over 60 years old is projected to nearly double from 12% to 22% between 2015 and 2050, increasing from 1 billion in 2020 to 1.4 billion by 2030 and nearly 2.1 billion by 2050 [1]. One of the common problems in the elderly population is accidental falls, which are common but sometimes life-threatening. The rising demand for fall detection systems, algorithms, and techniques has been evident in the surge of interest observed via Google Trends, a platform dedicated to monitoring internet users' information search patterns since 2004. Notably, the search term "fall detection" has attained an unprecedented peak, registering a remarkable surge of over 500% within the past five years [2].

There are approximately 6.1 million people in the United States who use mobility assistance devices, including walking canes, orthopedic walkers, and rollators. However, life-threatening falls in the older population are a crucial health and safety issue; about 1.5 million elderly people are injured by falling each year, and about 47,300 people per year using walking aids suffer injuries from falls that require an emergency room visit [3].

In addition, life-threatening falls often occur within the rehabilitation process after major surgery to the hip or leg area, which is an increasing concern for patients and medical experts alike. The risk of repeat falls is especially high in patients who have already sustained a hip fracture [4]. Thus, there is an urgent need for a smart device that can detect falls for orthopedic walker users in real time and alert caregivers for emergency assistance.

In recent years, wearable smart devices, such as Fitbit and Apple Watch, have become very popular for activity tracking and physiology monitoring; however, these smartwatches are not designed for walker users. Even if a walker user wears such a device on the wrist, the device may not accurately detect motions and hazards since both hands are rested on the walker during movement. In addition, some elderly users may not wear such devices reliably on their own. Traditional at-home health monitoring products often abandon the usage of tracking devices due to irregularities and false positives in favor of simple medical alert systems self-actuated by the subject, which do not consider the possibility that a fall may cause life-threatening injuries that may incapacitate the subject and cause them to be unable to activate such systems. Inhibitions to speech and capability of motion can render medical alert systems that rely on direct communication between emergency services and the user ineffective.

The primary objective of this research is to develop a novel, smart, and low-cost IoT device, namely SafeStride, that overcomes the limitations of the existing wearable and stationary fall detection systems. SafeStride is attached directly to an assistant walker, which continuously records and processes the walker's motion data in real time. SafeStride is intended to facilitate the development of a comprehensive fall detection system that efficiently tracks movement and promptly generates alerts in the event of a fall. In addition, it provides an activity log, such as walking steps and duration, standing time and stability, etc., to caregivers and physicians to help them assess a patient's progress with recovery and monitor if the patient has met the prescribed daily movement goals. This paper presents the detailed design and implementation of the hardware prototype, highlighting its potential advantages over existing solutions. Integrating this prototype into the walker holds promise for enhancing wearable fall detection capabilities, ultimately leading to improved safety and well-being of walker users.

The other objective of this research is to develop a dataset for assistive walker fall detection and motion classification. Machine learning is a powerful technique for data analysis tasks such as classification and detection. However, such a dataset for assistive walkers is unavailable in the literature. In this work, we performed data collection and labeling, followed by the evaluations of various machine learning models to validate their accuracy. Intuitively, a drastic fall-like motion that causes a user to lose balance or collapse to the ground usually results in significant spikes in acceleration on all three axes and noticeable deviations in gyroscopic position [5]. Similarly, acceleration and rotation data can also be used to detect the motion of intervals of walking and stopping. The primary challenge is to separate the motions of walking from falls while eliminating false positives and negatives. Missing a life-threatening fall is clearly critical, but false alarms can also seriously jeopardize the adoption of this new technology. Our labeled dataset is shared publicly on Kaggle [6] for easy access to the research community.

The main contributions of this work are the following: (1) We developed an IMU-based fall detection dataset for assistive walkers, which is published online for sharing with the IoT and the healthcare research community; (2) we evaluated different machine learning models and compared their performance using the dataset; (3) we built a low-power, real-time IoT device prototype and validated its functions through experiments. The IoT device can also communicate with others via Bluetooth and Wi-Fi, enabling it to transmit pertinent information to smartphones or a web server, facilitating prompt alerts in the event of falls. Additionally, the device collects and maintains various statistics, including the number of steps the user takes and their duration. These statistics can offer valuable insights to physicians who monitor a patient's progress in recovery.

In the following sections of this paper, the design, implementation, and evaluation of the proposed fall detection system will be presented. In Section 2, a review of the related work will be conducted for recognizing existing approaches and identifying gaps. The detailed design and implementation of the novel hardware prototype will be presented in Section 3. Section 4 will describe the procedure for data acquisition and labeling. Section 5 will discuss the machine learning and deep learning models to be used for fall detection.

The results and discussion of the SafeStride system evaluation will be included in Section 6. Finally, Section 7 will summarize the project, describe the main findings, and identify potential areas for future work.

2. Related Work

A wide range of approaches have been explored to design effective fall detection systems, including the use of wearable devices [7–11] and smartphones [12–15] with microphones [16], cameras [17], accelerometers and gyroscopes [18,19], GPS [20], and combinations of multiple sensors [21].

In addition to wearable devices, alternative approaches have been extensively investigated in fall detection systems. Depth sensors, such as Microsoft Kinect [22] or time-of-flight cameras [23], have been implemented with the primary goal of enhancing the accuracy and effectiveness of fall detection systems. Infrared sensors have been used to detect changes in infrared radiation within specific environments [24]. Doppler radar systems provide a non-invasive and privacy-preserving approach for timely and accurate fall detection among the elderly, analyzing unique time-frequency characteristics to identify fall events regardless of lighting conditions [25].

Vision-based methods analyze video data from cameras, employing techniques such as optical flow analysis [26], object tracking [27], and human pose estimation to detect falls based on changes in motion or posture [28]. Acoustic sensors, such as microphones or sound arrays, capture audio signals and employ signal processing techniques to identify sudden impact sounds, screams, or other acoustic patterns associated with falls [29].

Signal processing techniques play a crucial role in all of the aforementioned fall detection systems, as they extract meaningful features from sensor data, facilitating the accurate detection of fall events across diverse system types. Multiple signal processing techniques have been employed in fall detection systems, including orientation filters [9], quaternions [10], thresholding techniques [13,18–20], histograms of oriented gradients [17], clustering algorithms [21], Bayesian segmentation approaches [23], spatiotemporal analysis methods [27], Kalman Filters [30], sensor data fusion [31], and wavelet transforms [32].

Furthermore, machine learning methods have been applied extensively in recent years. In this field, a variety of machine learning models have been employed, ranging from simpler approaches, like decision trees [22] and k-nearest neighbors (kNN) [7,11,14,21,24,29], to more complex methods, such as Bayesian classifiers [23,25], support vector machines (SVM) [28], neural networks [16], and deep learning models [15,26].

A closely related research field is human activity recognition (HAR) by classifying human activities from motion sensor data. Several public datasets for fall detection using wearable sensors [33] are available. A dataset [34] in which data were labeled as fall, near fall, and activities of daily living (ADL) were most useful for us. Both the signal processing method [35] and the machine learning approach [5,36] are widely used in this field. In signal processing approaches, methods based on preset thresholds to detect a step [37], methods based on peak detection count steps by counting the peaks of sensor readings [38], and methods based on correlation analysis count steps by calculating and comparing the correlation coefficients between two neighboring windows of sensor readings [39]. In machine learning methods, authors design HAR algorithms based on convolutional neural networks (CNN) [40,41], and researchers in [36] have developed an algorithm based on long short-term memory (LSTM) to recognize human activities. It is worth mentioning that all existing HAR datasets were based on wearable IMU sensors directly attached to the human body. In contrast, this project involves fixing the sensor to a walker, so the data characteristics are quite different.

The research and development of fall detection systems specifically for assistive walkers and rollators is relatively limited. However, there have been several notable endeavors to create technology-integrated devices to improve the safety and functionality of these mobility aids.

In [42], a rollator-based ambulatory assistive device with an integrated non-obtrusive monitoring system was introduced, aiming to enhance the functionality and capabilities of traditional rollators. The Smart Rollator prototype incorporates multiple subsystems, including distance/speed measurement, acceleration analysis, force sensing, seat usage tracking, and physiological monitoring. Collected data are stored locally within a microprocessor system and periodically transferred to a remote data server via a local data terminal. This enables remote access and analysis of the data, contributing to improved monitoring and support for individuals using rollators.

The research work in [43] presents a fall detection system designed specifically for smart walkers. The system combines signal processing techniques with the probability likelihood ratio test and sequential probability ratio test (PLT-SPRT) algorithm to achieve accurate fall detection. Simulation experiments were conducted to identify the control model of the walker and analyze limb movements, while real-world experiments were performed to validate the system's performance.

The study in [44] demonstrates the experimental results of fall detection and prevention using a cane robot. Non-disabled male participants walked with the cane robot, and their "normal walking" and "abnormal walking" data were recorded. The fall detection rate was evaluated using the center of pressure-based fall detection (COP-FD) and leg-motion-based fall detection (LM-FD) methods. COP-FD detected falls by calculating the center of pressure (COP) during walking and comparing it to predefined thresholds. LM-FD used a laser range finder (LRF) to measure the relative distance between the robot and the user's legs, enabling the detection of leg motion abnormalities associated with stumbling. The results showed successful fall detection for both COP-FD and LM-FD, with some instances of false positives and false negatives.

While various machine learning techniques using sensor data have been explored for fall detection systems, a major obstacle in these projects is the lack of reliable data. Obtaining large and diverse datasets that capture different fall scenarios and environmental conditions is crucial for training robust models, posing a significant challenge that researchers and developers must address for effective machine learning-based fall detection systems.

The study in [33] presents a comprehensive analysis of publicly available datasets used for research on wearable fall detection systems. The study examines twelve datasets and compares them based on various factors, including experimental setup, sensor characteristics, movement emulation, and data format. The datasets primarily use accelerometers, gyroscopes, and magnetometers as the main sensors for capturing movement and orientation data. However, there is a lack of consensus and standardization among the datasets regarding the number and position of sensors, as well as the specific models and characteristics of the sensors employed. This heterogeneity makes it challenging to compare and evaluate fall detection systems effectively.

In summary, the current research on fall detection systems has focused primarily on wearable devices and stationary systems, with limited exploration of technology integration in walkers, rollators, canes, and wheelchairs. Integrating technology into walkers and rollators can improve fall detection capabilities and meet the specific needs of people who rely on these mobility aids. However, it is important to note that data availability remains a major challenge in this field, and creating reliable datasets is crucial to further progress in this research.

3. System Design and Hardware Prototype

This section provides an overview of SafeStride, a novel prototype designed as an add-on device for assistive walkers, as seen in Figure 1. SafeStride offers comprehensive fall detection, step counting, and activity tracking capabilities. The hardware design includes selecting components and their functionalities and the integration process.

An important feature of SafeStride is its onboard machine learning capability, enabling the device to run machine learning algorithms directly in real time. Using data from the inertial measurement unit (IMU) sensor, SafeStride analyzes movement patterns to detect

falls, steps, and other motions. As an IoT application, SafeStride can establish wireless connections with other devices, such as smartphones and web servers, enabling real-time alerts and intelligent monitoring. Moreover, healthcare professionals can access the data log to monitor a patient's recovery progress and provide personalized care.

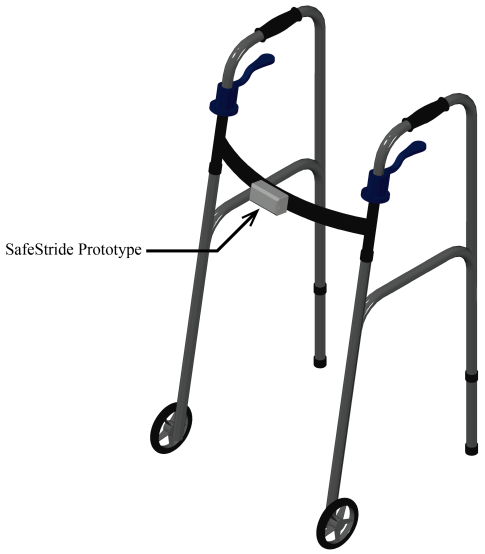

Figure 1. Visual depiction of a SafeStride prototype integrated into a walker device.

3.1. Microcontroller

Considering the requirement for machine learning capabilities, it might be assumed that using a powerful microprocessor is necessary. However, recent advancements have enabled the direct deployment of machine learning algorithms onto low-power microcontrollers [45]. There are cost-effective hardware development boards available off-the-shelf that integrate machine learning capabilities, low power consumption, IMU units, wireless connectivity, and a microphone into a single board, making them well suited for the SafeStride prototype.

The Arduino Nano 33 BLE Sense, which incorporates the nRF52840 microcontroller, LSM9DS1 IMU, MP34DT05 microphone, and BLE connectivity, is a prominent example of such a device [46]. It has been widely used in research experiments, demonstrating its effectiveness in various applications that demand machine learning capabilities, low power consumption, and integrated sensor functionalities [47–49]. Alternatively, the Seeed Studio XIAO nRF52840 microcontroller provides comparable capabilities [50]. During the development of this project, the dataset was developed using the Arduino Nano 33 BLE Sense board.

In this IoT application, the communication characteristics of the hardware components used are critical. Specifically, while the Arduino Nano is equipped with Bluetooth Low Energy, it does not support Wi-Fi wireless protocol. To address this limitation and enable wireless communication with both Bluetooth and Wi-Fi devices, as well as facilitate cloud communication with a web server, the ESP8266 has been incorporated as the main core. This component handles wireless communication by integrating Bluetooth and Wi-Fi functionalities. The nRF52840 microcontroller on the Arduino Nano board, meanwhile, is used for sensor data sampling and machine learning classification tasks. As illustrated in Figure 2, this configuration enables extensive wireless communication functions, paving the way for seamless connectivity and cloud-based interactions across multiple devices.

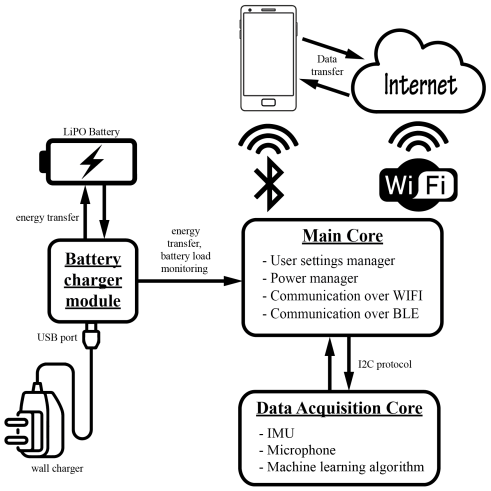

Figure 2. Graphical representation of the elements incorporated into this design. Main Core is a ESP8266 board and Data Acquisition Core is based on nRF52840.

3.2. Power Source and Protective Casing

The SafeStride fall detection system is designed to operate on a LiPO battery, which was chosen due to its high energy density, lightweight construction, and rechargeable capability. The LiPO battery provides the necessary power capacity to support the device's operation for extended periods without frequent recharges. To facilitate charging, a charging circuit is incorporated using power from a micro-USB port. In addition, a boosting circuit is employed to elevate the battery's voltage from 3.7 volts to the 5 volts required to power the circuit. It is important to note that the logic levels used within the circuit remain at 3.3 volts.

To ensure the protection and integration of the system components, a PCB design has been developed. The components are carefully planned to be integrated and enclosed within a 3D-printed case, providing secure and robust housing. This protective case is specifically designed to withstand potential impacts from falls and protect the electronics inside. Special precautions are taken to securely fasten the case to the walker using Velcro straps to ensure stability during use.

The prototype communicates with the server using the HTTPS protocol, ensuring secure and encrypted data transfer. The system's integration of both Wi-Fi and BLE enhances versatility and offers a fallback communication method. If the Wi-Fi connection has problems, the prototype can pair with smartphones through BLE to relay data to the server via the cellular network. This design choice ensures reliable and secure data transmission.

4. Building the Walker Dataset

A significant challenge arises in this particular research due to the lack of data available for the proposed IMU-based fall detection method. As a result, it is necessary to create a dataset that includes IMU data samples, covering various scenarios, including accidental falls and normal daily activities when using the walker, to train and evaluate a machine learning model. The data generated during this research are compiled into the "Walker Dataset" that is published on Kaggle [6]. The dataset serves as an open-source tool for sharing with other researchers in this field.

4.1. Sampling Setup

The sampling setup in the fall detection prototype focused on wireless communication and power autonomy. The microcontroller's built-in BLE capability enabled the wireless transmission of sensor data to a computer, eliminating the need for cables and allowing

unrestricted movement during data collection. Real-time communication with the computer was established using a Python script and BLE protocols.

To ensure precise capture of motion and orientation information, the sampling rate was optimized to achieve the highest possible data rate, reaching approximately 100 samples per second. This high sampling rate enabled a detailed analysis of movement patterns. The acquired sensor data, which included XYZ acceleration and gyroscope measurements, was stored in JSON format on the computer, simplifying subsequent parsing and extraction during the data processing stages.

4.2. Data Acquisition

For the IMU-based fall detection algorithm, the data acquisition process involves capturing motion and orientation data using the sensor integrated into the device. In this project, the fall detection problem is treated as a classification task, where four classes are considered: idle, motion, step, and fall.

- Idle: The idle class represents periods of no movement or minimal activity when the walker is stationary, and the user is not actively engaged in any motion.
- Motion: The motion class captures random motion, which means that the walker is being moved, but not necessarily by someone actively walking with it.
- Step: The step class corresponds to the specific movement when someone takes a step with the walker. This class focuses on capturing the gait pattern associated with normal walking using the walker.
- Fall: The fall class represents situations where the walker falls to the floor, indicating a potential fall event of the walker user.

Data for this study were collected using the walker in five groups. Each group contains about 100 to 150 samples of fall, motion, and step data.

To record idle data, multiple 5-minute duration files were recorded with the walker placed in various positions and kept still throughout the recording. The walker was placed in different orientations to capture variations in sensor readings, ensuring a diverse dataset.

For motion data, random movements were recorded by actively moving the walker in different directions and at varying speeds. This allowed for capturing a wide range of motion patterns, simulating everyday movements that may occur between adjacent walking steps.

To simulate the walking of an elderly person, walker data were recorded in an indoor open space. Pauses were introduced between each step to mimic the typical walking pattern of an elderly person and facilitate the separation of steps.

For falls, deliberate actions were performed in which the walker was intentionally thrown to the floor multiple times and then picked up again. Additionally, external forces, such as pushing the walker, were applied to simulate the motion associated with a fall event. These intentional actions aimed to capture the distinct patterns and characteristics of falls in the acquired sensor data.

4.3. Training Data Labeling

After recording the data on the computer, several steps were taken to process it. The first step involved downsampling the data from approximately 100 Hz to 80 Hz. This adjustment was necessary because the original sampling rate was not consistent, and the acceleration and gyro data were received separately, each with its own timestamp. By downsampling the data to a lower rate, interpolation was applied to ensure a fixed time delay of 12.5 ms between samples, corresponding to a sampling rate of 80 Hz. This process effectively synchronized the gyro and acceleration data, resulting in six dimensions for each timestamp.

Once the sampling rate was normalized and the data dimensions unified, the next step was to separate the groups of steps within the recorded data. In the experimental setup, each recording file contained multiple clusters of steps mixed with some turning

movements. Typically, 20 to 25 steps forward were taken, followed by a turn and then another series of walking steps.

To facilitate this separation, a Java interface was developed that allows step groups to be manually identified and isolated from turning movements, ensuring that each group is extracted individually for further analysis and classification. The interface provides the ability to set a lower and upper threshold, as seen in Figure 3, allowing information between the defined ranges to be exported to a new, separate JSON file. This feature allows the extraction of specific segments of logged data, providing more focused datasets for further analysis and processing.

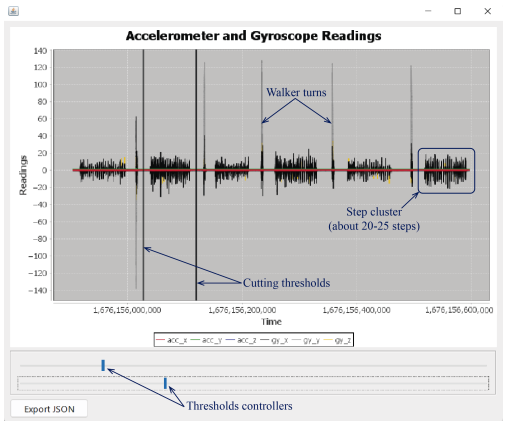

Figure 3. The Java interface is a user-friendly graphical tool designed for identifying and separating step groups from turning movements.

An algorithm was developed to extract and analyze individual steps after isolating groups of steps. This algorithm uses signal processing and machine learning techniques to process the recorded data. To create a single motion metric, accelerometer and gyroscope data were combined using the root mean square (RMS) calculation. This involves normalizing the data and calculating the square root of the average of the squared values, providing a concise representation of the overall motion intensity for each timestamp.

After RMS calculation and normalization, a running window averaging filter with 50 samples was applied to smooth the data. This filter reduces noise and jitter in the signal, resulting in a smoother representation of the motion intensity over time. Subsequently, the dataset was analyzed using a hidden Markov model (HMM) algorithm to identify hidden states. The HMM algorithm is a statistical model that analyzes sequential data with unobservable states. In this context, it distinguishes between steps and non-steps based on the intensity of movement captured from the RMS values, as observed in Figure 4.

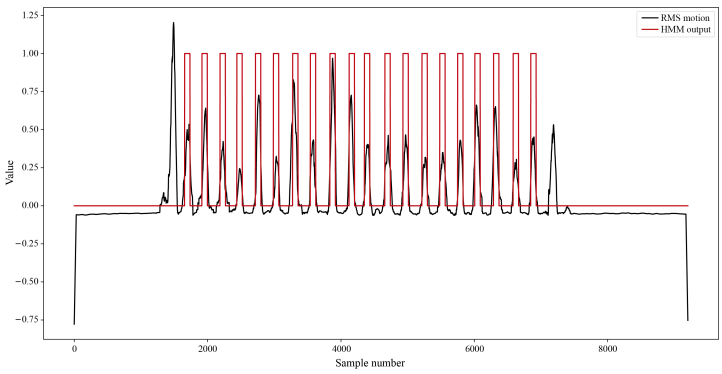

Figure 4. Root mean square motion values and hidden Markov model observed states, representing detected steps.

Step identification using the HMM algorithm allows for various analyses and measurements. It can determine the average length of a step by analyzing the step patterns identified in the dataset. This information provides valuable insight into the characteristics of the steps recorded during the experiment and allows the selection of an appropriate window size.

By selecting the window size based on the average length of steps recorded in the dataset, it is possible to capture the correct number of features needed to represent a step, optimizing classification accuracy and adhering to microcontroller limitations. Choosing a window size that is too short can result in insufficient information for accurate step identification, while a window size that is too long could consume too much memory and exhaust the processing capabilities of the microcontroller.

After analyzing the step lengths in the recorded data, a window size of 160 samples was chosen for further processing. Using this window size, all steps were extracted by calculating the centroid of each step, as determined by the HMM algorithm. The centroid represents the center of the step pattern in the time domain and provides a reference point for extracting the step segment. To extract the step sample, 80 samples to the left of the centroid and 80 samples to the right were selected, ensuring that each step was captured within a total of 160 samples. By extracting the step segments in this way, individual JSON files were created for each step, resulting in 600 unique and independent step samples extracted from the logged data. The visual representations of these data samples, as well as samples from other classes, can be seen in Figure 5.

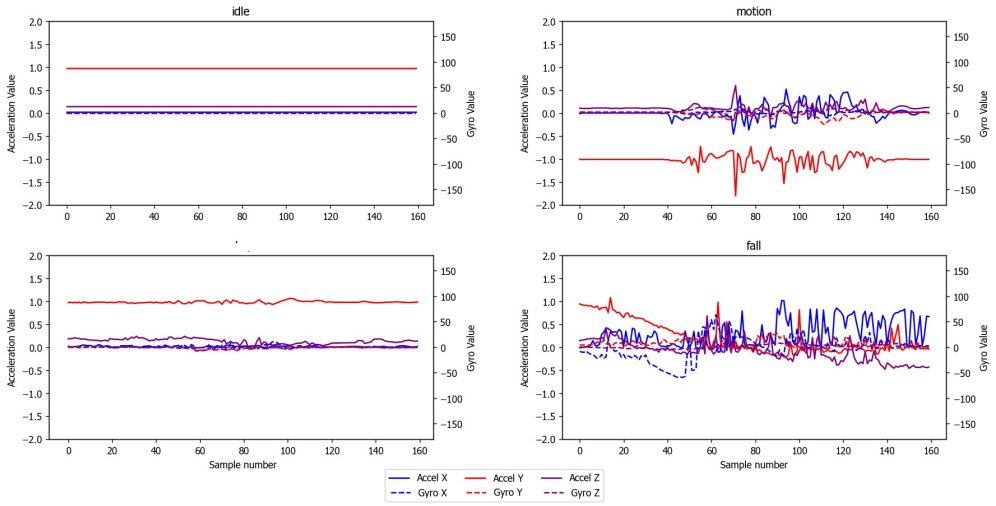

Figure 5. Visual representations of data samples from the dataset, showcasing distinct characteristics of each class.

Similarly, for the idle and motion classes, 620 samples were randomly selected from the raw files that were recorded during the data acquisition phase. The same procedure described for the steps was applied, using the HMM algorithm to isolate the falls from the recorded data. This resulted in the creation of 620 fall samples.

All extracted samples were consolidated into a unified dataset. Included in this dataset is a label column with the corresponding class for each sample, which could be “steps”, “idle”, “motion”, or “fall”. In addition, 960 different features are included, with all six dimensions (XYZ acceleration and XYZ gyroscope), and 160 samples contributed to each dimension. Table 1 shows the detailed schema of the dataset, indicating the column names, data types, units, and a brief description for each.

Table 1. Dataset schema for the inertial measurement unit-based walker motion recording.

Column Name	Data Type	Units	Description
Activity_Type	String	-	Class label ("idle", "motion", "step", or "fall")
Acc_{x_1} to $Acc_{x_{160}}$	Float	m/s^2	Acceleration data for X-axis (160 samples)
Acc_{y_1} to $Acc_{y_{160}}$	Float	m/s^2	Acceleration data for Y-axis (160 samples)
Acc_{z_1} to $Acc_{z_{160}}$	Float	m/s^2	Acceleration data for Z-axis (160 samples)
$Gyro_{x_1}$ to $Gyro_{x_{160}}$	Float	deg/s	Gyro data for X-axis (160 samples)
$Gyro_{y_1}$ to $Gyro_{y_{160}}$	Float	deg/s	Gyro data for Y-axis (160 samples)
$Gyro_{z_1}$ to $Gyro_{z_{160}}$	Float	deg/s	Gyro data for Z-axis (160 samples)

5. Machine Learning Classification

IMU-based fall detection is a widely explored approach that leverages the motion and orientation data captured by inertial measurement units to detect fall events. By analyzing the patterns and characteristics of the IMU data, it becomes possible to identify sudden changes or anomalies that may indicate a fall. This section outlines a study of different machine learning and deep learning algorithms for assistive walker fall detection and motion classification using the walker dataset that we developed in Section 4.

To identify the most suitable classification algorithm for our problem, we implemented a systematic grid search strategy using the Scikit-Learn set of classifiers, initially exploring a wide range through the all_estimators tool. This approach not only allowed us to evaluate the broad landscape of classification algorithms but also ensured that we did not overlook any potential high-performing candidates.

For a more rigorous evaluation, we employed a k-fold cross-validation technique for each algorithm, performing 10 rounds of training and testing processes. Throughout this evaluation, we not only considered accuracy but also considered other essential performance indicators, including precision, recall, and F1 score, ensuring a holistic assessment of each classifier's capability.

From this evaluation, the top 10 classifiers, based on their average accuracy, were shortlisted for further analysis. It is worth noting that these algorithms were initially tested with their default parameters. In this machine learning search process, we further improved the classification performance with dedicated hyperparameter tuning.

We also explored the performance of deep learning using a six-layered convolutional neural network (CNN) that was adapted from the classic Lenet-5 architecture in image classification. The CNN architecture is shown in Figure 6. The input of the CNN network is 160 points of 6-axis data collected by the IMU sensor, so the input feature size is 160-by-6. Data were first fed through four convolution layers used for feature extraction. Each layer encompasses 2-D filters for convolution, data pooling, and reLU functions. Data then enter two fully connected layers and a softmax layer. The CNN produces the four-class classification results as the output.

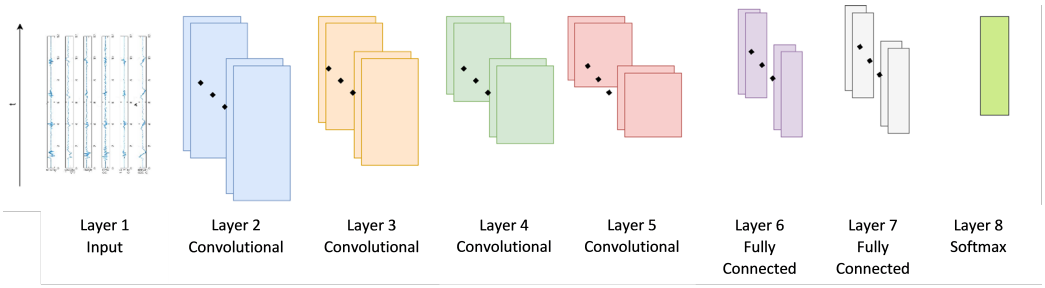

Figure 6. Architecture of a simple convolutional neural network for motion classification.

6. Results, Implementation, and Testing

This section presents the results of evaluating the aforementioned classification models using the dataset that we developed in this work. Subsequently, the system was tested in a lab environment, and it successfully detected falls and steps for walker users.

6.1. Machine Learning Classification Results

Applying the grid search on the existing machine learning models available in Scikit-Learn, we found 10 models with the best overall classification performance. In Table 2, the performance results of the top 10 algorithms, evaluated by the above metrics, were documented.

Table 2. Performance metrics of multiple machine learning classifiers for the inertial measurement unit dataset.

Machine Learning Model	Accuracy	Precision	Recall	F1-Score
Random Forest Classifier	98.59%	98.62%	98.59%	98.58%
Hist Gradient Boosting Classifier	97.79%	97.80%	97.78%	97.78%
Extra Trees Classifier	97.71%	97.72%	97.71%	97.70%
Bagging Classifier	95.56%	95.63%	95.56%	95.55%
Extra Tree Classifier	92.58%	92.62%	92.58%	92.52%
Decision Tree Classifier	90.48%	90.57%	90.48%	90.45%
KNeighbors Classifier	89.31%	91.09%	89.31%	89.28%
NuSVC	88.83%	91.72%	88.83%	88.26%
Gaussian NB	88.59%	90.63%	88.59%	87.63%
Logistic Regression CV	87.30%	88.70%	87.30%	86.73%

The Random Forest Classifier stood out as the most effective model in our evaluations. To better understand its performance, we split the dataset: 80% was used for training and the remaining 20% for testing. Considering the stochastic nature of the Random Forest algorithm, we ran it 100 times to ensure we captured a consistent measure of its capabilities. In all these runs, it achieved a maximum accuracy of 98.59%.

Figure 7 presents the confusion matrix with the results of applying the trained algorithm to the test dataset. The model showed good performance on the fall detection task. There was no missed fall event nor false positive of fall classification, which is critically important to this product. However, occasional failures were observed when the algorithm attempted to discriminate between motion and steps, attributable to the similar inertial characteristics presented by these classes. Overall, the RandomForestClassifier model demonstrated superior performance.

As for the proposed six-layer deep learning model, the CNN was trained using the same walker dataset to perform the motion classification task. Then, 80% of the samples were randomly selected for training, and the remaining 20% of samples were used for validation. The model was trained for 10 epochs. The results show that CNN reached an overall classification accuracy of 97.4%, which is lower than the Random Forest classifier. Although many high-performance, deeper CNN models may be trained to reach superior accuracy using the same dataset, the microcontroller on the hardware prototype can only handle a small CNN with a few layers and limited parameters due to the constraint of processing power and memory. We attempted to use deeper models using the TinyML library to generate the C code for implementation. But, it failed to load into the microcontroller during the execution, exceeding its usable memory limit.

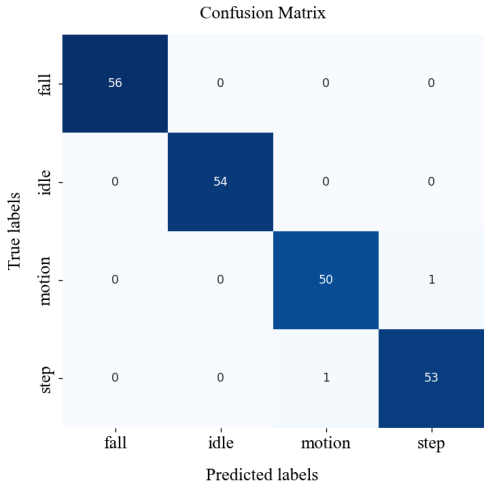

Figure 7. Confusion matrix with the performance of the Random Forest Classifier on the test dataset.

6.2. Software Implementation

From the study of the machine learning and deep learning classifiers, we chose the Random Forest model for embedded software implementation. To implement the trained machine learning model in a microcontroller environment, the next step is to convert the model to an executable C code file. This conversion was performed using the m2cgen Python library [51], which transforms a trained machine learning model into a format compatible with microcontrollers. By generating the C code file, the model can be uploaded to the microcontroller, enabling it to perform classification tasks directly on the device. The generated code file, totaling 1.2 MB in size (approximately 25,500 lines of code), is fully compatible with microcontrollers without causing any concerns regarding available flash memory capacity.

The input for the trained model consists of 160 samples, each capturing data from six dimensions: accelerometer and gyroscope data in X, Y, and Z. These samples are directly obtained from the built-in IMU within the microcontroller board. Combining the six dimensions results in a total of 960 features, which are used as input for the trained model.

Once the model is uploaded to the microcontroller, it can process the input data received from the IMU and execute the machine learning C code to obtain an output. This output corresponds to the classification of the sampled movement as idle, motion, step, or fall, providing real-time fall detection capabilities directly from the microcontroller. The fall detection system can operate autonomously without relying on external computational resources. This integration enables efficient and prompt classification of movement data, enhancing the overall performance and responsiveness of the fall detection system.

6.3. Real-Time Testing with Continuous Data

Continuous, unseen data were used to test the IMU-based detection method, aiming to replicate a real-time setup. Integrating the machine learning model into the microcontroller of the SafeStride system facilitated the experimentation with features transmitted to the MCU via serial communication. To ensure efficient data processing, a graphical interface was developed, enabling the loading and visualization of continuous samples and their subsequent transmission to the controller, as can be seen in Figure 8. The implementation employed a sliding window technique, employing a window size of 160 samples with a 50% overlap. Subsequently, the data were received by the microcontroller, classification was performed, and the corresponding result was returned. By comparing the recorded and transmitted data, highly favorable outcomes were obtained.

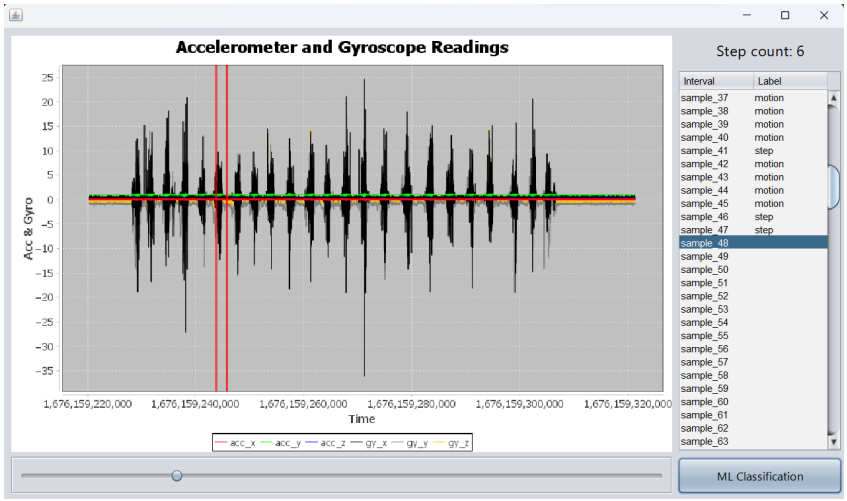

Figure 8. A graphical interface was developed to test the Safestride prototype with continuous data. Recorded data were loaded and sent to the prototype via Serial Communication, utilizing a sliding window (indicated by red vertical lines). The prototype performed machine learning classification using its trained model and promptly returned the result to the application (see the table on the right side).

In contrast to traditional machine learning testing methods, this experimental setup distinguishes itself by employing continuous data instead of pre-processed and curated static samples. Within this framework, various critical aspects were comprehensively evaluated, including accurate step detection, mitigating missteps or indications of falls arising from random motion, and the immediate triggering of fall events when they occur. Real-time laboratory tests demonstrated the satisfactory performance of the trained algorithm, thereby confirming the robustness of both the model and the training dataset. Additionally, this experiment convincingly demonstrated the feasibility of implementing complex machine learning algorithms on an IoT device, particularly for tasks involving real-time and continuous data processing.

6.4. Additional Redundancy

For the purpose of redundancy and enhanced reliability, we take advantage of an existing microphone on the Arduino Nano board by implementing a keyword spotting function on SafeStride as an extra layer of safety measures if the walker user falls. The keyword spotting fall detection technique was based on the Cyberon Speech Recognition Engine, which provided the ability to download a pre-trained machine learning model from its web platform for easy, fast, and effective implementation on microcontrollers such as the Arduino Nano board. This technique is used in speech recognition systems to detect predefined keywords in continuous speech. In the context of fall detection, it can be adapted to detect verbal expressions commonly uttered during or immediately after a fall, such as exclamations of surprise, pain, or calls for help. The use of keyword spotting for fall detection offers the potential advantage of not only detecting a fall event but also capturing the urgency or severity of the fall. For instance, if the system detects phrases like “Help” or “Help me”, it can provide additional evidence of a fall event and potentially trigger more urgent assistance.

In the initial phase of this project, an attempt was made to create a machine learning model using a publicly available dataset for keyword spotting; however, a dataset that addressed the specific contextual requirements for detecting words or phrases potentially uttered during a fall event could not be identified. Creating such a dataset is very difficult due to the inherent variability observed in pitch, accent, speed, and other speech characteristics. To overcome these challenges, the decision was made to utilize the built-in library of the Cyberon speech recognition engine. The engine-generated model was tested in a lab

environment, where its ability to identify the specific keyword "help" was demonstrated. Keyword spotting was included as a supplementary tool in case the motion classification algorithms did not capture a rare fall event.

7. Conclusions and Future Work

This research presents SafeStride as an IoT device targeted at fall detection and activity logging of assistive walkers by applying machine learning on the IMU sensor data. A walker dataset is developed and shared with the public. Various machine learning algorithms are evaluated using the dataset, and the Random Forest classifier is among the best, which achieves an overall accuracy of 98.59% with no missed falls. The SafeStride system prototype is tested in the lab environment, covering a wider range of fall scenarios. It has demonstrated accurate and reliable performance in detecting falls, counting steps, and motion classification in real time. Additionally, the IoT device connects with cloud resources through wireless communication, further expanding its capabilities and potential for prompt response if a fall event occurs.

For future work, one promising avenue is to explore the integration of other sensing modalities into the SafeStride system, such as cameras, radars, pressure sensors, etc., to create a more robust and comprehensive fall detection system. Moreover, user feedback and field testing should be pursued to validate the system in real-world scenarios involving various types of falls and environmental conditions.

An important contribution of this work is the creation of the walker dataset shared on Kaggle [6]. This dataset serves as a valuable resource for training and evaluating machine learning and deep learning models for fall detection. The availability of this dataset to the public also encourages further research and development in the field, thereby driving progress in fall detection and enhancing care for the elderly and individuals with physical disabilities. Overall, the SafeStride system represents a significant step forward in the field of fall detection, harnessing the power of machine learning to provide a more reliable and responsive solution. The results obtained are very encouraging and demonstrate the potential for new advances in IoT for healthcare.

Author Contributions: Conceptualization, M.H.; methodology, M.H. and A.G.; software, M.H. and A.G.; validation, M.H. and A.G.; formal analysis, M.H.; investigation, A.G.; resources, A.G.; writing, M.H. and A.G.; visualization, A.G.; supervision, A.G.; project administration, A.G. All authors have read and agreed to the published version of the manuscript.

Funding: A. Garcia is supported by the Fulbright-SENACYT Scholarship.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset is available at https://www.kaggle.com/dsv/6493939, accessed on 18 September 2023.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
CNN	Convolutional Neural Network
BLE	Bluetooth Low Energy
IMU	Inertial Measurement Unit

References

1. WHO Ageing and Health. 2023. Available online: https://www.who.int/news-room/fact-sheets/detail/ageing-and-health (accessed on 18 September 2023).
2. Wang, X.; Ellul, J.; Azzopardi, G. Elderly fall detection systems: A literature survey. *Front. Robot. AI* **2020**, *7*, 71. [CrossRef]
3. CDC Falls Are Leading Cause of Injury and Death in Older Americans. 2016. Available online: https://www.cdc.gov/media/releases/2016/p0922-older-adult-falls.html (accessed on 18 September 2023).
4. Weil, T.P. Patient falls in hospitals: An increasing problem. *Geriatr. Nurs.* **2015**, *36*, 324–347. [CrossRef]
5. Chelli, A.; Pätzold, M. A Machine Learning Approach for Fall Detection and Daily Living Activity Recognition. *IEEE Access* **2019**, *7*, 38670–38687. [CrossRef]
6. Gonzalez, A.G. Walker Fall Detection Dataset. 2023. Available online: https://www.kaggle.com/dsv/6493939 (accessed on 18 September 2023).
7. de Quadros, T.; Lazzaretti, A.E.; Schneider, F.K. A Movement Decomposition and Machine Learning-Based Fall Detection System Using Wrist Wearable Device. *IEEE Sens. J.* **2018**, *18*, 5082–5089. [CrossRef]
8. Majumder, S.; Mondal, T.; Deen, M.J. Wearable Sensors for Remote Health Monitoring. *Sensors* **2017**, *17*, 130. [CrossRef] [PubMed]
9. Pierleoni, P.; Belli, A.; Palma, L.; Pellegrini, M.; Pernini, L.; Valenti, S. A high reliability wearable device for elderly fall detection. *IEEE Sens. J.* **2015**, *15*, 4544–4553. [CrossRef]
10. Wu, F.; Zhao, H.; Zhao, Y.; Zhong, H. Development of a wearable-sensor-based Fall Detection System. *Int. J. Telemed. Appl.* **2015**, *2015*, 1–11. [CrossRef] [PubMed]
11. Attal, F.; Mohammed, S.; Dedabrishvili, M.; Chamroukhi, F.; Oukhellou, L.; Amirat, Y. Physical human activity recognition using wearable sensors. *Sensors* **2015**, *15*, 31314–31338. [CrossRef] [PubMed]
12. Sousa Lima, W.; Souto, E.; El-Khatib, K.; Jalali, R.; Gama, J. Human activity recognition using inertial sensors in a smartphone: An overview. *Sensors* **2019**, *19*, 3213. [CrossRef]
13. Cao, Y.; Yang, Y.; Liu, W. E-FallD: A fall detection system using Android-based smartphone. In Proceedings of the 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, Chongqing, China, 29–31 May 2012; pp. 1509–1513. [CrossRef]
14. Tsinganos, P.; Skodras, A. A smartphone-based fall detection system for the elderly. In Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis, Ljubljana, Slovenia, 18–20 September 2017; pp. 53–58. [CrossRef]
15. Hassan, M.M.; Gumaei, A.; Aloi, G.; Fortino, G.; Zhou, M. A Smartphone-Enabled Fall Detection Framework for Elderly People in Connected Home Healthcare. *IEEE Netw.* **2019**, *33*, 58–63. [CrossRef]
16. Cheffena, M. Fall Detection Using Smartphone Audio Features. *IEEE J. Biomed. Health Inform.* **2016**, *20*, 1073–1080. [CrossRef] [PubMed]
17. Ozcan, K.; Velipasalar, S. Wearable Camera- and Accelerometer-Based Fall Detection on Portable Devices. *IEEE Embed. Syst. Lett.* **2016**, *8*, 6–9. [CrossRef]
18. He, Y.; Li, Y.; Bao, S.D. Fall detection by built-in tri-accelerometer of smartphone. In Proceedings of the 2012 IEEE-EMBS International Conference on Biomedical and Health Informatics, Hong Kong, China, 5–7 January 2012; pp. 184–187. [CrossRef]
19. Rakhman, A.Z.; Nugroho, L.E.; Widyawan; Kurnianingsih. Fall detection system using accelerometer and gyroscope based on smartphone. In Proceedings of the 2014 The 1st International Conference on Information Technology, Computer, and Electrical Engineering, Semarang, Indonesia, 8 November 2014 ; pp. 99–104. [CrossRef]
20. Fung, N.M.; Wong Sing Ann, J.; Tung, Y.H.; Seng Kheau, C.; Chekima, A. Elderly Fall Detection and Location Tracking System Using Heterogeneous Wireless Networks. In Proceedings of the 2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE), Kota Kinabalu, Malaysia, 27–28 April 2019; pp. 44–49. [CrossRef]
21. Tsinganos, P.; Skodras, A. On the comparison of wearable sensor data fusion to a single sensor machine learning technique in fall detection. *Sensors* **2018**, *18*, 592. [CrossRef] [PubMed]
22. Stone, E.E.; Skubic, M. Fall Detection in Homes of Older Adults Using the Microsoft Kinect. *IEEE J. Biomed. Health Inform.* **2015**, *19*, 290–301. [CrossRef]
23. Diraco, G.; Leone, A.; Siciliano, P. An active vision system for fall detection and posture recognition in elderly healthcare. In Proceedings of the 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010), Dresden, Germany, 8–12 March 2010; pp. 1536–1541. [CrossRef]
24. Chen, W.H.; Ma, H.P. A fall detection system based on infrared array sensors with tracking capability for the elderly at home. In Proceedings of the 2015 17th International Conference on E-health Networking, Application & Services (HealthCom), Boston, MA, USA, 14–17 October 2015; pp. 428–434. [CrossRef]
25. Wu, Q.; Zhang, Y.D.; Tao, W.; Amin, M.G. Radar-based fall detection based on Doppler time-frequency signatures for assisted living. *IET Radar Sonar Navig.* **2015**, *9*, 164–172. [CrossRef]
26. Hsieh, Y.Z.; Jeng, Y.L. Development of Home Intelligent Fall Detection IoT System Based on Feedback Optical Flow Convolutional Neural Network. *IEEE Access* **2018**, *6*, 6048–6057. [CrossRef]

27. Lee, Y.S.; Lee, H. Multiple object tracking for fall detection in real-time surveillance system. In Proceedings of the 2009 11th International Conference on Advanced Communication Technology, Gangwon, Republic of Korea, 15–18 February 2009; pp. 2308–2312.
28. Huang, Z.; Liu, Y.; Fang, Y.; Horn, B.K.P. Video-based Fall Detection for Seniors with Human Pose Estimation. In Proceedings of the 2018 4th International Conference on Universal Village (UV), Boston, MA, USA, 21–24 October 2018; pp. 1–4. [CrossRef]
29. Li, Y.; Ho, K.C.; Popescu, M. A Microphone Array System for Automatic Fall Detection. *IEEE Trans. Biomed. Eng.* **2012**, *59*, 1291–1301. [CrossRef]
30. He, J.; Bai, S.; Wang, X. An unobtrusive fall detection and alerting system based on Kalman filter and Bayes network classifier. *Sensors* **2017**, *17*, 1393. [CrossRef]
31. Li, H.; Shrestha, A.; Fioranelli, F.; Le Kernec, J.; Heidari, H.; Pepa, M.; Cippitelli, E.; Gambi, E.; Spinsante, S. Multisensor data fusion for human activities classification and fall detection. In Proceedings of the 2017 IEEE SENSORS, Glassboro, NJ, USA, 13–15 March 2017; pp. 1–3. [CrossRef]
32. Palmerini, L.; Bagalà, F.; Zanetti, A.; Klenk, J.; Becker, C.; Cappello, A. A wavelet-based approach to fall detection. *Sensors* **2015**, *15*, 11575–11586. [CrossRef]
33. Casilari, E.; Santoyo-Ramón, J.A.; Cano-García, J.M. Analysis of Public Datasets for Wearable Fall Detection Systems. *Sensors* **2017**, *17*, 1513. [CrossRef]
34. Ojetola, O.; Gaura, E.; Brusey, J. Data set for fall events and daily activities from inertial sensors. In Proceedings of the 6th ACM Multimedia Systems Conference, Portland, OR, USA, 18–20 March 2015; Association for Computing Machinery: New York, NY, USA, 2015; MMSys'15, pp. 243–248. [CrossRef]
35. Gu, F.; Khoshelham, K.; Shang, J.; Yu, F.; Wei, Z. Robust and Accurate Smartphone-Based Step Counting for Indoor Localization. *IEEE Sensors J.* **2017**, *17*, 3453–3460. [CrossRef]
36. Ordóñez, F.J.; Roggen, D. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* **2016**, *16*, 115. [CrossRef]
37. Link, J.B.; Smith, P.; Viol, N.; Wehrle, K. FootPath: Accurate map-based indoor navigation using smartphones. In Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation, Guimaraes, Portugal, 21–23 September 2011; pp. 1–8. [CrossRef]
38. Zhang, H.; Yuan, W.; Shen, Q.; Li, T.; Chang, H. A Handheld Inertial Pedestrian Navigation System with Accurate Step Modes and Device Poses Recognition. *IEEE Sens. J.* **2015**, *15*, 1421–1429. [CrossRef]
39. Pan, M.S.; Lin, H.W. A Step Counting Algorithm for Smartphone Users: Design and Implementation. *IEEE Sens. J.* **2015**, *15*, 2296–2305. [CrossRef]
40. Ha, S.; Choi, S. Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 381–388, ISSN: 2161-4407. [CrossRef]
41. Zeng, M.; Nguyen, L.T.; Yu, B.; Mengshoel, O.J.; Zhu, J.; Wu, P.; Zhang, J. Convolutional Neural Networks for human activity recognition using mobile sensors. In Proceedings of the 6th International Conference on Mobile Computing, Applications and Services, Austin, TX, USA, 6–7 November 2014; pp. 197–205. [CrossRef]
42. Chan, A.D.C.; Green, J.R. Smart Rollator Prototype. In Proceedings of the 2008 IEEE International Workshop on Medical Measurements and Applications, Ottawa, ON, Canada, 9–10 May 2008; pp. 97–100. [CrossRef]
43. Ding, D.M.; Wang, Y.G.; Zhang, W.; Chen, Q. Fall Detection System on Smart Walker Based on Multisensor Data Fusion and SPRT Method. *IEEE Access* **2022**, *10*, 80932–80948. [CrossRef]
44. Di, P.; Hasegawa, Y.; Nakagawa, S.; Sekiyama, K.; Fukuda, T.; Huang, J.; Huang, Q. Fall Detection and Prevention Control Using Walking-Aid Cane Robot. *IEEE/ASME Trans. Mechatron.* **2016**, *21*, 625–637. [CrossRef]
45. Warden, P.; Situnayake, D. *Tinyml: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*; O'Reilly Media: Sebastopol, CA, USA, 2019.
46. Arduino. Arduino Nano 33 BLE Sense. 2023. Available online: https://store-usa.arduino.cc/products/arduino-nano-33-ble-sense (accessed on 18 September 2023).
47. Waqar, D.M.; Gunawan, T.S.; Morshidi, M.A.; Kartiwi, M. Design of a Speech Anger Recognition System on Arduino Nano 33 BLE Sense. In Proceedings of the 2021 IEEE 7th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA), Bandung, Indonesia, 23–25 August 2021; pp. 64–69. [CrossRef]
48. Wardhany, V.A.; Subono; Hidayat, A.; Utami, S.W.; Bastiana, D.S. Arduino Nano 33 BLE Sense Performance for Cough Detection by Using NN Classifier. In Proceedings of the 2022 6th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), Yogyakarta, Indonesia, 13–14 December 2022; pp. 455–458. [CrossRef]
49. Trivedi, K.; Shroff, H. Identification of Deadliest Mosquitoes Using Wing Beats Sound Classification on Tiny Embedded System Using Machine Learning and Edge Impulse Platform. In Proceedings of the 2021 ITU Kaleidoscope: Connecting Physical and Virtual Worlds (ITU K), Geneva, Switzerland, 6–10 December 2021; pp. 1–6. [CrossRef]

50. Moon, G.B. Seeed Studio XIAO nRF52840 Sense-TinyML/TensorFlow Lite-IMU/Microphone-Bluetooth5. 2023. Available online: https://www.seeedstudio.com/Seeed-XIAO-BLE-Sense-nRF52840-p-5253.html (accessed on 18 September 2023).
51. BayesWitnesses. Bayeswitnesses/m2cgen: Transform ML Models into a Native Code (Java, C, Python, Go, JavaScript, Visual Basic, C#, R, PowerShell, PHP, Dart, Haskell, Ruby, F#, Rust) with Zero Dependencies. 2023. Available online: https://github.com/BayesWitnesses/m2cgen (accessed on 18 September 2023).