





Fall Detection Using Intelligent Walking-Aids and Machine Learning Methods

Thanh-Nghi Doan^{1,2}(✉) , Eliane Schroeter³(✉), and Thanh-Binh Phan^{1,2} 

¹ Faculty of Information Technology, An Giang University, An Giang, Vietnam
{dtngchi, ptbinh}@agu.edu.vn

² Vietnam National University, Ho Chi Minh City, Vietnam

³ Jade-Hochschule, University of Applied Science, Wilhelmshaven, Germany
eliane.schroeter@student.jade-hs.de

Abstract. Walking aids are commonly given to older adults to prevent falls, but paradoxically, their use has been identified as a risk factor for falling, which is a prevalent issue among this population, causing serious injuries, disabilities, and even death. This has resulted in a significant increase in public health care and the development of remote health monitoring technology to enhance home care devices. One of the key issues being addressed is the identification of falling incidents, which can aid in the rapid arrival of assistance and prevent additional harm. This paper is to develop intelligent walking aids using machine learning methods and an M5Stack Core2 microcontroller. These aids analyze information from various sensors such as accelerometers and gyroscopes to identify falls. Touch and location sensors are also utilized to determine the device's usage and location in case of an emergency. The collected data is sent to a web server in JSON format via the M5Core2's WiFi module, allowing for a quick response if necessary. The fall detection system has been extensively tested, resulting in 99.62% accurate identification of falls.

Keywords: Fall detection · Remote health monitoring · Machine learning · M5Stack Core2 · Intelligent walking-aid

1 Introduction

Detecting and preventing falls is crucial, especially for older adults who are more susceptible to falls due to aging-related factors. Home care devices, including fall detection systems, have become essential in ensuring the safety of the elderly and individuals with chronic illnesses who wish to remain at home. This leads to the development of a novel system that incorporates Intelligent walking aids to detect falls and enhance user safety and quality of life. The device provides an added layer of security, allowing users to have confidence in their mobility and engage in activities while ensuring their safety. Over the past years, fall detection and supervision systems have been classified into various categories.

One of the widely adopted categories is wearable devices, which are worn on the body and equipped with a variety of sensors placed in different areas. Since they are worn all the time, they offer continuous monitoring and tracking of overall activities, making them relatively reliable. The authors in [1] suggested a strong activity recognition method for smart healthcare that involves utilizing body sensors and a deep convolutional neural network (CNN). Various body sensors used in healthcare, such as ECG, magnetometer, accelerometer, and gyroscope sensors, are analyzed to extract significant features from the signal data. The feature extraction process involves Gaussian kernel-based principal component analysis and Z-score normalization. To evaluate the effectiveness of the proposed approach, a publicly available standard dataset is used, and the results are compared to other conventional approaches. The authors in [2] proposed an approach for fall detection that employs three distinct sensors placed at five different locations on the subject's body to gather data for training purposes. The UMAFall dataset is utilized to obtain sensor readings and train models for fall detection. Five different models are trained, each corresponding to one of the five sensor models, and a majority voting classifier is employed to determine the final output. However, for some patients, the constant wearing of these devices depicts an inconvenience or, especially when talking about elderly people, they simply forget to put their intelligent devices on.

In contrast to wearable systems, vision-based systems offer a comfortable and reliable option for fall detection in-home care. The authors in [3] provided a brief overview of vision-based fall detection, outlining recent methods and highlighting their advantages and disadvantages, and touching on possible future research topics. The authors in [4] examined the latest non-intrusive (vision-based) fall detection techniques based on deep learning (DL). They also provide an overview of benchmark datasets for fall detection and explain various metrics used to assess the performance of these systems. The authors in [5] introduced a novel smart camera system for real-time monitoring, recognition, and remote warning of abnormal patient actions. The proposed method is cost-effective and easy to deploy. It does not require ambient sensors and uses regular video camera footage for detection. The system utilizes high-precision human body pose tracking with MediaPipe Pose and employs the Raspberry Pi 4 device and LSTM network for real-time classification and remote monitoring of patient actions. One of the key advantages of these approaches is their ability to differentiate a person's actions without requiring them to wear a device. However, they are limited to a single space and can intrude on a person's privacy. For example, a vision-based device cannot be installed in private areas like the bathroom where falls can also occur. Additionally, some individuals may feel uncomfortable being continuously monitored, despite the data protection measures that are inherent in these devices.

The proposal of using an environmentally based system to address privacy concerns in fall detection. Existing attempts with infrared sensor systems and millimeter-wave radar sensors have limitations but offer the advantage of not

being worn and can be installed in private spaces. The paper introduces an intelligent walking aid that combines environmental solutions and wearable devices. By adding sensors to the walking aid, it can detect falls and alert caregivers. The device restores mobility, provides autonomy, increases confidence and safety, and cannot be forgotten. It preserves privacy and can be used anywhere the walking aid is taken. The cost is affordable, and the device focuses on walking activities without the need to differentiate other activities. It primarily aims to improve safety and mobility over long distances. The device can be combined with other fall-detection measures, and further studies can be conducted to gather data on falls with walking aids. The main contributions of this paper consist of:

- Build a comprehensive detection system with a user-friendly interactive interface that can immediately monitor the usage and orientation of the walking aid.
- Create a novel dataset containing over 255 data sets collected from a fall study or real-life scenario.
- Develop a fall detection algorithm for a walking aid that uses touch as input. This algorithm will be implemented on an M5Stack Core2 microcontroller, which will analyze sensor data to detect falls.
- Develop a machine learning model for fall prediction and create a website that is compatible with it. The goal is to interpret the data collected by the M5Core2 through WiFi.

The overview of our proposed system is described in Fig. 1. There are two kinds of walking aids that are frequently used and have been under consideration for this particular situation. These aids are illustrated in Fig. 2. The first aid, shown on the left (a), is a commonly used walking cane. On the right (b) is a walking aid that stands on four feet, making the process of detecting a falling person a bit more complicated.

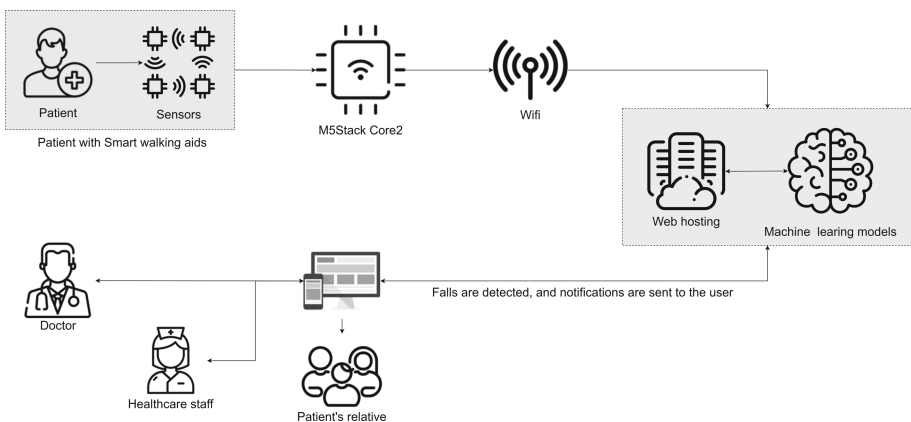


Fig. 1. The overview of our proposed system of fall detection.



Fig. 2. Two common walking aids: (a) Walking cane, (b) Four-legged walking aid

2 Materials and Methods

2.1 M5Stack Core2

To control the sensors and process data, we utilize an M5Stack Core2, which is equipped with an ESP32 microcontroller (Fig. 3). The device can be charged and connected to other devices with ease through its USB Type-C interface. It comes with an inbuilt lithium battery, featuring a capacity of 390 mAh, and an AXP192 power management chip. The battery level can be checked using the small LED lamp located on the side of the device, which can be turned on or off by pressing the power button located on the side of the device. A reset button is present on the bottom of the device. The M5Stack Core2 features a two-inch touch screen, which is ideal for user interaction. Three capacitive buttons located at the bottom of the screen can also be programmed. Additionally, the M5Stack Core2 has a WiFi module and supports Bluetooth, making wireless data transfer seamless. The device features dual-core Xtensa 32-bit 240 Mhz LX6 processors that can be controlled independently. It also comes with 16 MB Flash memory and 8MB PSRAM. Other features include a vibration motor, an RTC module for precise timing, a 6-axis IMU sensor, a microphone, an SD card slot, and a loudspeaker. The M5Core2 is compatible with various programming platforms, including the Arduino IDE, UIFlow, and MicroPython. The inbuilt sensors can be extended by utilizing numerous external sensors compatible with the M5Core2, which can be connected using cables and intersections if required.

Previously, M5Stack devices have been employed for medical supervision, such as in the case of an M5StackGrey utilized to create a motion game that encourages elderly individuals to remain active and flexible through exercise [6]. The device communicates wirelessly with a computer and features a small compatible vibration motor that can be controlled. In [7], the authors utilized



Fig. 3. M5Stack Core2 ESP32 IoT Development Kit.

an affordable and easy-to-set-up ESP32-CAM Module by M5, equipped with an OV2640 Wireless Camera Module, for a fall warning system based on visual cues. In a similar vein, the authors in [8] proposed a fall detection system that employs infrared sensors and transfers data wirelessly via WiFi using an M5Stack device. Additionally, the authors in [9] employed the M5Stack Core to create a walking aid for individuals with visual or motor impairments, with the device serving as a sensor data collector and interpreter that sends feedback to a mobile application through vibration alerts when obstacles are present or the path is unclear. The M5Core2 device is particularly relevant to this study due to its built-in gyroscope and accelerometer, which enable the detection of changes in orientation and acceleration and aid in fall identification. Additionally, the device's WiFi module is crucial for transmitting data to a web server. Further sensors capable of identifying touch can be connected to the M5Core2 through cables, as discussed in Sect. 2.2. Lastly, the loudspeaker and touchscreen functionalities are also important to this work, particularly with respect to the alarm tone and interactive interface.

2.2 Sensors

The M5Stack Core2's built-in MPU6886, which includes both an accelerometer and a gyroscope, is employed to detect falls. Data can be obtained through either the M5Core2's WiFi module or via Bluetooth. While the WiFi option is a straightforward way to wirelessly transmit data, it is reliant on a stable internet connection and may be subject to local limitations. Alternatively, data can be sent to a phone using Bluetooth, but this approach requires the user to carry the phone at all times and ensure that it is not misplaced. To determine the usage of the walking aid, additional external sensors were considered. Since the sensor unit is fixed to a crutch, it is necessary to use sensors that can detect if the walking aid is currently being used. Various potential sensors were explored, such as heart rate monitors, temperature sensors, and distance measuring sensors. Ultimately, the solution that was chosen is the Adafruit VL6180X, a proximity sensor that can be easily attached to the handle without requiring physical contact. This sensor can measure objects up to a distance of 10 cm and can be installed at the beginning of the handle to detect if a hand is placed on the handle or not. By detecting if an object is placed before it, the sensor can reliably determine if the walking aid is currently being used. This sensor can be

integrated with the main code and all its features. In testing the fall detection algorithm, only the acceleration sensor and gyroscope data were utilized to conserve resources. Nevertheless, a reflective sensor or a sensor capable of measuring the distance to the ground could also be considered as an additional feature for fall detection. These sensors could be installed at the handle, but their inclusion would affect the power consumption of the device. Therefore, this possibility may be examined in the future or used as a supplement if the gyroscope and accelerometer data prove to be unreliable over time. Finally, to determine the user's location in case of a falling event, a GPS sensor can be integrated. The M5Stack Atom GPS Sensor is selected for this purpose. With its addition, help can be directed to the right location as soon as possible.

2.3 Sending M5Stack Core2 Data to a Web Server

The process of sending M5Stack Core2 data to a web server is described in Algorithm 1. It is written in the programming language C++ with the Arduino IDE and consists of a number of different functions, which together form the final version of the code.

Algorithm 1 Sending M5Core2 data to a web server

```

if Not Found WiFi configuration on Flash then
    Set up WiFi
end if
Connect to WiFi
if Not Found UUID on Flash then
    Create a UUID and write it on Flash.
end if
Get the UUID of M5Core2.
if The proximity sensor detects the usage of the handle then
    The device keeps on taking data from the sensor.
else
    Enter a sleep mode to save energy.
end if
Create a JSON file consisting of sensor data.
Get data acceleration sensor and Gyroscope.
if Fall detected (GyroY<0.8) then Turn on the Alarm.
end if
if Help/Doctor button was pressed then
    Send the JSON file to the Web server immediately.
else
    Turn off the Alarm.
end if
Send the JSON file to the Web server.

```

The fundamental requirement for the fall detection method is a functional WiFi connection to the M5Core2. Without an active internet connection, data

transfer is impossible. Therefore, the first step is to verify the availability of an internet connection, preferably one that the device has previously connected to. If the M5Core2 finds the necessary configuration information for the WiFi network in the flash memory, it automatically connects to the network and proceeds to the next stage of the fall detection code. In the absence of any saved WiFi information in the flash memory, the device prompts the user to provide a new WiFi name and password, which are then stored in the flash memory for future use. Once an internet connection is established, the sensor data can be collected and added to the data that will be sent in JSON format, as required.

To ensure that the web hosting can accurately identify the device sending data, it's important to assign a unique identification to each M5Stack Core2. This is achieved by generating a Universally Unique Identifier (UUID) for each device, which is stored in its flash memory. Once generated, the UUID remains the same and is used for all subsequent data transmissions, ensuring the device is correctly paired with its corresponding patient.

The following step is to verify if the device is being used. As previously mentioned, this is accomplished by monitoring the handle using a proximity sensor. If the sensor detects handle usage, the device will continue to gather sensor data. However, if the sensor does not detect usage for an extended period of time, the device will enter sleep mode to save energy and wake up when touch is detected again.

The code also includes an interactive interface with three buttons displayed on the screen. Pressing the “Doctor” or “Help” button will trigger the alarm and send a JSON file to the web hosting to request help immediately. The “Okay” button, on the other hand, will turn off the alarm and send a message to the host indicating that no help is needed if a fall has been detected by the device.

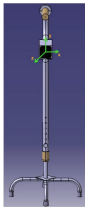
3 Results and Discussion

3.1 Data Collection

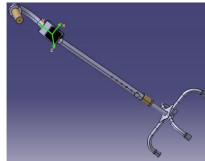
In order to estimate fall risk, several existing methods such as [10–15] have been studied and used to generate fall-related datasets for their research purposes. However, these datasets were not suitable for our system’s hardware architecture. Therefore, a new dataset was created for this study to accurately determine the occurrence of falls. The primary tool for detecting falls is the MPU6886 3-axis-acceleration sensor and 3-axis-gyroscope, which is built into the M5Core2. While the accelerometer measures acceleration in all directions, the gyroscope detects the device’s orientation and the attached walking aid. Each gyroscope value ranges from 1 to -1, indicating the current position of the crutch. Since the sensor is not attached to the human body but to the walking aid, the methodology for fall detection needs to be reevaluated. To conduct the experimental phase, we need to differentiate between three states in which the crutch can fall together with the user.

The first state that needs to be differentiated is the upright position of the crutch, which occurs when it is not in use when the patient is standing, and

when the person is walking. Therefore, smaller impacts on the acceleration and gyroscope data cannot be classified as falls and must be labeled as “normal position.” Any actions classified as “normal position” are considered harmless and do not require any action from the system. The upright position is characterized by a y -value of the gyroscope that is close to 1, while the x and z values should be around 0. Since the crutch will bend slightly while the person is walking, small variations are allowed. The forward movement can be observed in the y -value of the acceleration data. While elderly people are generally expected to move more slowly than younger individuals, rapid movements can still occur when they are adjusting their walking aid for the next step. Therefore, it is crucial to monitor both the orientation and acceleration data to determine whether the combination suggests a normal position or a fall. A visual representation of the walking aid in an upright or normal position can be seen in Fig. 4a.



(a) Normal position with the coordinate system.



(b) Falling position according to the coordinate system.



(c) Fallen position according to the coordinate system.

Fig. 4. Walking aids position according to the coordinate system

The second state pertains to the falling process, where the crutch loses its upright position and continues to fall for a certain period. Detecting a fall is crucial and significant changes in the gyroscope and acceleration data are key indicators for this state, which is classified as “fall detected”. A fall is considered harmful and triggers a message from the device to the web hosting, indicating that assistance might be necessary if no further action is taken. The data gathered during a fall can vary widely, as it can occur in various situations, and the crutch’s orientation can differ greatly depending on the fall’s direction and speed. Accurately interpreting the combination of sensor data is vital for detecting a fall. During a fall, the acceleration data can rapidly increase to triple digits. The y - and z -values are the most affected when falling to the left or right, while the x - and y -values show the greatest acceleration during a forward or backward fall. At the same time, the crutch’s orientation changes from a vertical to a horizontal position. During a fall, the y -value of the gyroscope drops from about 1 to 0, while the x - or z -value changes to 1 or -1 , depending on the fall’s direction. If the crutch falls to the right, the x -value drops towards -1 , while it increases to 1 if it falls to the left. A forward fall is indicated by the z -value climbing up to 1, whereas a rapid decline to -1 indicates a backward fall. Figure 4b shows the walking aid’s orientation during a fall, based on the coordinate system.

The final stage is when the crutch is in a fallen position, which indicates that a fall has occurred, and the walking aid is lying still on the floor. This state is labeled as “fallen” and is characterized by little to no variation in the acceleration and gyroscope data. Although it is not considered harmful, it may serve as an indicator of whether the person using the crutch was able to stand up again. The crutch is deemed to have fallen when there is no acceleration detected and the gyroscope values suggest that the crutch is in a horizontal position (y-value close to 0 and one of the other values around 1 or -1). A depiction of the fallen walking aid in the coordinate system can be found in Fig. 4c.

In the experimental phase of the study, a significant amount of data was required to be gathered while using the walking aid in different states, including the upright position, the fallen position, and a simulated falling scenario. The collected data was then organized and stored in tables to train the algorithm to accurately distinguish between the three states. To achieve this, various walking patterns were simulated, and data was collected for the “normal position.” Moreover, the crutch was made to fall under different conditions and in various directions, and the position of the crutch after each fall was recorded as data.

To collect the first set of falling data, a walking cane was used, and the M5Core2 was fixed on it. A total of over 257 sets of data were collected, divided into three categories: “normal position”, “fall detected”, and “fallen”, and stored in a table. The table includes all the necessary data from the acceleration sensors and gyroscopes, such as the x, y, and z values of the gyroscope and the x, y, and z values of the acceleration data, along with the corresponding label. Several examples of the collected data are shown in Table 1. In order to determine whether a dataset should be labeled as “fallen”, “fall detected” or “normal position”, the considerations outlined earlier must be taken into account. When the crutch is in an upright or “normal position”, the y-value of the gyrosensor will typically be around one. However, during walking, the crutch may be bent, resulting in the y-value dropping to values of around 0.8. Therefore, the label “normal position” is assigned to data where the y-value of the crutch is above 0.8, considering the angle that may be applied while walking.

Should the walking aid fall, the y-value of the gyroscope will drop from what is considered upright into what can be identified as a falling event. Data, where the y-value exceeds 0.8 and a significant acceleration, can be noted is labeled as “fall detected”. The walking aid is considered to be falling as long as the acceleration can be identified and the y-value of the gyro-sensor varies between 0.1 and 0.8. When the walking aid falls, the gyroscope’s y-value drops from the upright position to indicate a falling event. Data that shows a y-value exceeding 0.8 and a significant acceleration is labeled as “fall detected”. The walking aid is considered to be falling as long as the acceleration can be detected and the y-value of the gyro-sensor varies between 0.1 and 0.8. Finally, the label “fallen” is assigned to the data when the walking aid is in a horizontal position and there is minimal acceleration detected, and the y-value of the gyro-sensor is near 0. This combination of sensor readings strongly suggests that the walking aid has fallen and is no longer in an upright position. To use the four-legged walking aid,

Table 1. Representative samples of collected falling data with a walking cane.

No	Label	AccX	AccY	AccZ	GyroX	GyroY	GyroZ
1	normal position	0,7935	13,9160	8,8501	-0,0862	1,0085	0,0388
2	normal position	-17,4561	39,4287	20,9961	-0,2188	1,0361	0,0764
3	normal position	-1,0986	81,4819	10,4370	-0,1743	0,9155	-0,1189
4	fall detected	217,2241	-10,4370	34,6679	-0,0234	-0,1667	-0,3433
5	fall detected	131,8969	17,3950	13,0005	0,0442	0,5579	-0,1936
6	fall detected	-141,6016	27,4658	22,5830	0,0718	0,5652	0,2709
7	fallen	0,2441	14,3433	11,9019	-0,9963	0,1245	-0,0508
8	fallen	0,7935	14,2212	12,207	-0,0525	-0,0145	-0,9446
9	fallen	0,5493	12,9395	10,7422	0,8945	-0,0452	-0,3535

a second dataset was collected, comprising a total of 255 data sets. These were classified into three categories: “normal position”, “fall detected”, and “fallen”.

3.2 Evaluation Metrics

Our performance metrics are precision, recall, and F1-score. Before defining the metrics, we define True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) terms. TP is the number of correctly classified samples by the algorithm. TN is the number of correctly missed samples. FP is the number of wrongly detected samples. FN is the number of wrongly missed samples. The TN is not used in calculating our performance metrics. Precision was the one factor that was used for the main comparison, as it is a good indicator of how reliable a method works on a specific data set. It mainly focuses on the identification of data as true positive or false positive and calculates how accurately a method can predict a falling event of the crutch. It is calculated with Formula 1. The recall metric measures the number of wrongly missed samples by the algorithm. Its value decreases as the number of false negatives increases. Recall is an important evaluation criterion in the applications that necessitate no misses in the classification task. It is computed as the ratio of correctly detected objects to the total number of samples (both correctly classified and missed), the formula of recall is in Formula 2.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}} \quad (2)$$

3.3 Applying the Machine Learning Method

In order to analyze the data on the web hosting, a classification problem needs to be solved in order to label the received data correctly to “normal position”,

“fall detected”, and “fallen”. Therefore it needs to be evaluated, what machine learning method is most suitable for the specific problem by cross-validation. The recorded data gets split into training and testing data. Most of the data, depending on the cross-validation method, is used to train the algorithm, whereas the rest of the data is then used to test how reliable the algorithm works on the problem. This is done for every method, to find the best one to use. In this specific case, the training data consisted of 2/3 of the collected data and the testing data consisted of the other 1/3. A 5-folded cross-validation method was used and the process was repeated ten times for each method. In this study, the performance of Support Vector Machine [16], Neural Network [17], and Gradient Boosting [18] models was assessed on a dataset collected from a walking cane.

Using a Cost (C) configuration of 1.00, a Regression loss epsilon (ϵ) of 0.10, a Kernel RBF, a numerical tolerance of 0.0010, and an iteration limit of 100. In this case, our data is six-dimensional, as we have six different types of information that are processed. The Acceleration data in the X, Y, and Z-axis and the data of the gyroscope in the X, Y, and Z-axis. Support Vector Machine finds these Support Vector Classifiers using kernel functions, namely polynomial or radial kernels, and moves data into a higher dimension. The values used for the kernel functions are determined by cross-validation. In order to test SVM on this set of data a radial Kernel was used with a regression loss ϵ of 0.1. As shown in Fig. 5, SVM achieves an accuracy of 100% for predicting 134 samples of the “fall detected” label, 97.40% for predicting the “normal position” label (75 samples), and 100% for predicting the “fallen” label.

Confusion matrix for SVM (showing number of instances)					
		Predicted			Σ
		fall detected	fallen	normal position	
Actual	fall detected	134	0	0	134
	fallen	0	46	0	46
	normal position	2	0	75	77
Σ		136	46	75	257

Fig. 5. Confusion matrix using SVM classifier.

Using the Neural Network machine learning method with a configuration of 100 neurons in the hidden layer, ReLu activation function, Adam optimization solver, regularization of 0.0001, and a maximum number of iterations of 200. In this case, the input nodes are the acceleration and gyroscope data and the output is the classification of the current position of the walking aid. As shown in Fig. 6, It achieves an accuracy of 99.25% for predicting 133 samples of “fall detected” label, 100% for predicting the “normal position” label (77 samples), and 100% for predicting the “fallen” label (46 samples).

Using the Gradient Boosting machine learning method with a configuration of 100 trees, a learning rate of 0.1, a maximum depth limit of 3 for individual trees, and a minimum subset size for the splitting of 2, and using a fraction of

Confusion matrix for Neural Network (showing number of instances)					
		Predicted			Σ
		fall detected	fallen	normal position	
Actual	fall detected	133	1	0	134
	fallen	0	46	0	46
	normal position	0	0	77	77
Σ		133	47	77	257

Fig. 6. Confusion matrix using Neural Network classifier.

1.00 of the training instances, As shown in Fig. 7, It achieves an accuracy of 99.25% for predicting 133 samples of “fall detected” label, 98.70% for predicting the “normal position” label (76 samples), and 97.82% for predicting the “fallen” label (45 samples).

Confusion matrix for Gradient Boosting (showing number of instances)					
		Predicted			Σ
		fall detected	fallen	normal position	
Actual	fall detected	133	0	1	134
	fallen	1	45	0	46
	normal position	1	0	76	77
Σ		135	45	77	257

Fig. 7. Confusion matrix using Gradient Boosting classifier.

3.4 Numerical Results and Discussion

In order to decide, which machine learning method was suited the best for this data set, the confusion matrices were taken into consideration and a ROC analysis was conducted. The default threshold for the ROC analysis was set to a typical 0.5. It shows the rate between True Positives (TP), meaning correctly identified cases of the wanted attribute, and False Positives (FP), falsely classified data as the wanted attribute. As only very few data sets were falsely classified, the graph is drawn very close to one.

The confusion matrices the ROC analysis is based on can be seen in Figs. 5 to 7, which clearly shows how each method performed on the given data set. Figure 5 shows, that the Support Vector Machine falsely predicted two cases as a “fall detected”, when in fact they should have been classified as normal positions. As shown in Fig. 6, Neural Network falsely classified a fall as an already fallen walking aid. While Naive Bayes and Gradient boosting show a few more errors with some confusion among all classes (Fig. 7). k-NN performed quite well again, classifying two falls falsely as a normal positions. The overall performance of the methods compared to one another by different parameters can be seen in Table 2. It compares the methods by AUC (Area under the curve), CA (Classification

accuracy), F1, Precision, and Recall. AUC compares the curves from the ROC analysis to detect which curve has the greatest area underneath. CA focuses on the proportion of the examples that were classified correctly.

Table 2. Comparison between the different machine learning methods in the percentage of correct identifications.

Machine Learning Method	AUC	Precision	Recall
Support Vector Machine	99.99	99.23	99.22
Neural Network	99.98	99.62	99.61
Gradient Boosting	99.34	98.84	98.83

After evaluating the methods using precision as the ultimate metric, it is clear that the Neural Network exhibited the most promising outcomes in detecting a falling event, with a reliability of 99.62%. On the other hand, Naive Bayes had the poorest result, with only 97.71% precision. Overall, the Neural Network model performed the best among all the methods, considering all the given parameters.

Once the data has been modeled and the most appropriate machine-learning method has been chosen, the resulting model can be deployed on the web hosting for further use. To accomplish this, the model must first be saved and then transferred using the pickle library. Once deployed on the web hosting, the model can analyze the data received from the M5 device and aid in identifying instances of falling.

3.5 Website and Message Distribution

The data is sent via a WiFi connection to the web host, where the patient's information is stored in a database, allowing the association between the device and the patient. Two scenarios are presented for fall detection: the first involves detecting falls within the device itself, which conserves energy but may have a slight risk of missing falling events during data transmission. The second scenario involves continuously sending and analyzing data using machine learning, providing more precise data collection at the expense of higher power consumption. The website stores various information, including patient data, device data, user group information, and contact lists. As shown in Fig. 8, the patient's data includes personal details, such as name, gender, date of birth, contact information, and address. Device data includes sensor information and a unique identifier for matching devices with patients. The website visually displays the collected data, and there are different sections for patient information, emergency contacts, and doctor information, each with specific privileges and information requirements based on their account types.

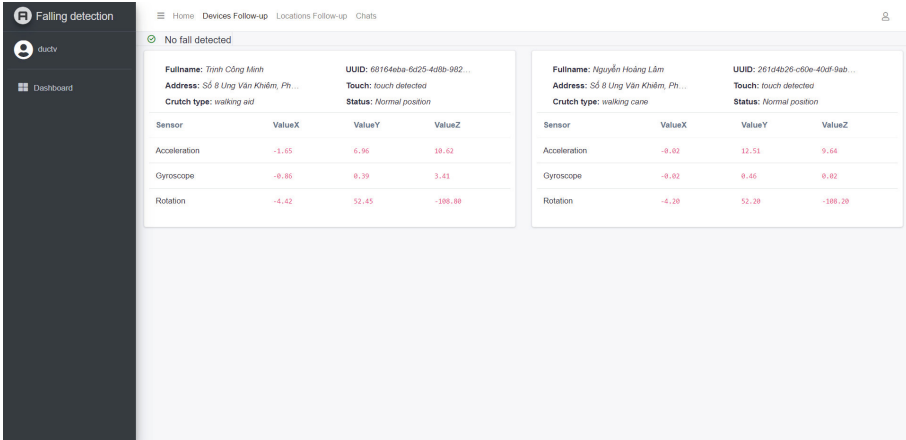


Fig. 8. Layout of the patient follow-up of the website.

4 Conclusion

The paper presented a novel fall detection system using an intelligent walking aid and machine learning methods. The system uses an M5Core2 microcontroller, the Adafruit VL6180X proximity sensor, and the MPU6886 3-axis acceleration and 3-axis gyro sensors to detect falls. An M5Stack Atom GPS sensor can also be used for tracking the current position of the device. The fall detection algorithm and overall system were tested with experiments, and the Neural Network machine learning method performed best, classifying 99.62% of the data correctly. Potential solutions could involve incorporating force sensors on the handle or feet of the walking aid or adding distance sensors to the stick to measure the distance to the legs. Alternatively, machine learning could be utilized to analyze walking patterns and sensor data to detect falls where the walking aid bounces back after impact.

Acknowledgment. The authors would like to thank to An Giang University, Jade University, and Vietnam National University in Ho Chi Minh City, Vietnam.

References

1. Uddin, M.Z., Hassan, M.M.: Activity recognition for cognitive assistance using body sensors data and deep convolutional neural network. *IEEE Sens. J.* **19**(19), 8413–8419 (2019)
2. Mankodiya, H., et al.: Xai-fall: explainable AI for fall detection on wearable devices using sequence models and XAI techniques. *Mathematics* (2022)
3. Kong, X.: Vision-based fall detection: methods and challenges. In: *2022 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pp. 157–160 (2022)

4. Alam, E., Sufian, A., Dutta, P., Leo, M.: Vision-based human fall detection systems using deep learning: a review. *Comput. Biol. Med.* 105626 (2022)
5. Doan, T.-N.: An efficient patient activity recognition using LSTM network and high-fidelity body pose tracking. *Int. J. Adv. Comput. Sci. Appl.* **13**(8) (2022)
6. Tatsukawa, S., et al.: Development of motion game for elderly based on sensory stimulus presentation. In *International Conference on Electrical, Computer and Energy Technologies (ICECET)*, pp. 1–4 (2021)
7. Sok, C., Saechan, C., Sittipong, C., Kaewmorakot, U., Poolperm, L.: Fall warning machine with image processing via the internet of things. In: *6th International STEM Education Conference (iSTEM-Ed)*, pp. 1–4 (2021)
8. Tateno, S., Meng, F., Qian, R., Hachiya, Y.: Privacy-preserved fall detection method with three-dimensional convolutional neural network using low-resolution infrared array sensor. *Sensors* **20**(20), 5957 (2020)
9. Grzeskowiak, F., Devigne, L., Pasteau, F., Dutra, G.S.V., Babel, M., Guégan, S.: Swalkit: a generic augmented walker kit to provide haptic feedback navigation assistance to people with both visual and motor impairments. In: *International Conference on Rehabilitation Robotics*, pp. 1–6 (2022)
10. Yoshida, T., et al.: A data-driven approach for online pre-impact fall detection with wearable devices, pp. 133–147 (2022)
11. Jia, N.: Detecting human falls with a 3-axis digital accelerometer. In: *Analog Dialogue* (2009)
12. Di, P., et al.: Fall detection and prevention control using walking-aid cane robot. *IEEE/ASME Trans. Mechatron.* **21**(2), 625–637 (2016)
13. Malasinghe, L., Ramzan, N., Dahal, K.: Remote patient monitoring: a comprehensive study. *J. Ambient. Intell. Humaniz. Comput.* **10**, 01 (2019)
14. Pan, D., Liu, H., Dongming, Q., Zhang, Z.: Human falling detection algorithm based on multisensor data fusion with svm. *Mob. Inf. Syst.* **1–9**(10), 2020 (2020)
15. Ramirez, H., Velastin, S.A., Meza, I., Fabregas, E., Makris, D., Farias, G.: Fall detection and activity recognition using human skeleton features. *IEEE Access* **9**, 33532–33542 (2021)
16. Cortes, C., Vapnik, V.: Support-vector networks. *Mach. Learn.* **20**(3), 273–297 (1995)
17. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. *Bullet. Math. Biophys.* **5**(4), 115–133 (1943)
18. Friedman, J.H. Greedy function approximation: a gradient boosting machine. *Annal. Statist.* 1189–1232 (2001)