Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2021.015589

ARTICLE

Improve the Accuracy of Fall Detection Based on Artificial Intelligence Algorithm

Ming-Chih Chen, Yin-Ting Cheng[*] and Ru-Wei Chen

Department of Electronic Engineering, National Kaohsiung University of Science and Technology (First Campus), Kaohsiung City, 82445, Taiwan

[*]Corresponding Author: Yin-Ting Cheng. Email: i107109118@nkust.edu.tw

Received: 29 December 2020 Accepted: 07 June 2021

ABSTRACT

This work presents a fall detection system based on artificial intelligence. The system incorporates miniature wearable devices for fall detection. Fall detection is achieved by integrating a three-axis gyroscope and a three-axis accelerometer. The system gathers the differential data collected by the gyroscope and accelerometer, applies artificial intelligence algorithms for model training and constructs an effective model for fall detection. To provide easy wearing and effective position detection, it is designed as a small device attached to the user's waist. Experiment results have shown that the accuracy of the proposed fall detection model is up to 98%, demonstrating the effectiveness of the model in real-life fall detection.

KEYWORDS

Artificial intelligence; fall detection; miniature sensing device

1 Introduction

1.1 Research Motivation

The current reduction in birth rate and extended life expectancy have resulted in the ageing of the population, which has become a global issue. According to [1], the aged population will increase drastically in the future, the percentage of the aged population will continue to rise within the global population and reach 28% by the year 2050.

Falling is an important health issue for the senior population as it may cause serious harm to the elderly [2]. If an elderly person falls, he or she may not be able to move or regain consciousness and can only wait passively for medical assistance. However, if adequate assistance is not provided within a short time frame, more serious harm may result. Delays in receiving treatment may cause difficult or irrecoverable damages and increase the cost and burden of health care [3,4]. Additionally, some elderly people may develop fears of falling again and become limited in their range of movements, thus decreasing their quality of life. Therefore, the instantaneous fall detection of the elderly in an indoor environment and the establishment of an effective caring system are crucial for the aged population.

1.2 Research Purpose

We propose a system based on artificial intelligence for fall detection to reduce manual cost and provide early warnings. The system's goal is to detect the current positions of personnel within a monitored range and determine if a falling event has occurred. The current behaviours of the personnel are returned to the monitoring system for centralized record and management to establish the conditions of indoor activities of the personnel. The proposed system integrates a three-axis gyroscope, a three-axis accelerometer, as well as a Bluetooth module for wireless communication to design a waist-attached, miniature fall detection device. The device collects information from the gyroscope and accelerometer for analysis to deduce continuous signals representing the human body postures. According to the posture and signal relationship, artificial intelligence is used to construct a highly accurate model. Information associated with falling is transmitted by the waist-attached, miniature fall detection device via Bluetooth wireless packets to a receiving end designed using a Raspberry Pi computer and collected. The information is transmitted through the built-in local area network in the Raspberry Pi to the computer management interface to facilitate decision making after a fall.

1.3 Literature Review

The current fall detection system implementation methods can be divided into two categories: fixed fall detection system and non-fixed fall detection system. The fixed fall detection system [5] is mainly based on image recognition detection, floor vibration detection, wireless signal detection methods and so on. The system detects whether or not the user has fallen through sensors built on the wall or floor. The advantage of the system is that the user does not need to wear additional equipment and the disadvantages are that it is location-specific and the cost is high. The non-fixed fall detection system [6–9] mainly uses components such as the RFID, gyroscope, accelerometer and so on to detect whether or not a user has fallen via sensors attached to the user's body. The advantages of this system are that the computational complexity and the cost are low, however, the disadvantage is that the accuracy is relatively low.

In 2014, Wang et al. [5] proposed a fall detection system based on wireless network signals. This method determines if a user has fallen according to the current physical movement of a user, which is calculated via the differences between the transmitted signal and the received signal within the Wi-Fi equipment installed in the environment. This approach effectively reduces issues regarding equipment installation and privacy invasion since there is no need to install additional equipment if there is an existing Wi-Fi device in the environment.

In 2017, De Miguel et al. [8] presented a low-cost fall detector for smart homes based on artificial vision algorithms. The proposed detector combines several algorithms as input into a machine learning algorithm. The detector is able to provide high detection accuracy of fall diction.

In 2018, Shahzad et al. [10] proposed a fall detection system based on a three-axis sensor on a smartphone. The proposed approach attaches the phone to the user's waist or leg and uses the three-axis acceleration values to construct a fall model with a set threshold and a multiple kernel learning support vector machine (MKL-SVM). The model is integrated with a mobile application to calculate the user's current status. The model constructed using machine learning algorithms achieves an accuracy of 97.81% when attached to the waist and 91.70% when attached to the legs without requiring additional equipment.

In the same year, Ichwana et al. [11] proposed an accelerometer-based fall detection system. In the proposed system, a sensing module is attached to the waist, and a user's fall is determined by the user's movements, which are identified according to the calculated angular and vertical velocities. This method has low computational complexity and makes quick decisions.

In 2020, Nooruddin et al. [12] proposed a system based on client-server architecture. The system can be implemented using any type of IoT devices with internet connectivity and can be interfaced with four modules. The developed linear classifier model used in this system has achieved 99.7% accuracy in fall detection.

In the same year, Clemente et al. [13] presented a smart system performing fall detection based on floor vibration data produced by fall downs. Only using floor vibration as the recognition source, the system incorporates a person identification through vibration produced by footsteps to inform who is the fallen person. It is able to detect fall downs with an acceptance rate of 95.14%.

2 Research Method

Nowadays, artificial intelligence is widely used in various systems. This research uses TensorFlow as the framework for building a neural network (NN) and uses three different neural network architectures to build a fall detection model. The purpose of the neural network is to find the optimal weight value and deviation, which are obtained when the input data passes through the function in the framework. Therefore, an activation function is added after the hidden layer of the neural network to facilitate the finding of the desired result and calculate the suitable value to build a predictive model for fall detection. This work uses the multi-layer perceptron (MLP), recurrent neural network (RNN) and long short-term memory (LSTM) models to compare their advantages and disadvantages as shown in Tab. 1.

Table 1: Comparison of the advantages and disadvantages of neural networks

Neural network	Advantages	Disadvantages
MLP	Simple structure Solves classification problems	Difficult to select the number of nodes. Learning speed is slow. Suffers from the overfitting problem.
RNN	Able to handle continuous signals	Suffers from the vanishing gradient or exploding gradient problems. Signal is poorly processed for a long time.
LSTM	Continuous processing Long-term signals	Suffers from the exploding gradient problem. Complex architecture. Time-consuming calculation.

2.1 Multi-Layer Perceptron (MLP)

The MLP is a kind of feedforward neural network with three layers: an input layer, a hidden layer and an output layer. The input layer is the current material to be learned. The hidden layer is the feature node that needs to be learned and each node is a neuron with a nonlinear activation function so that its output meets the required results. The output layer is the category that needs to be learned. Each layer in the middle is a fully connected layer. The multi-layer perceptron was widely used in the 1980s in applications such as speech recognition, image recognition, machine translation and so on, and achieved good results in classification problems.

2.2 Recurrent Neural Network (RNN)

The early neural network architecture had no time concept and labeling learning for sequence signals. In 1982, Hopfield [14] proposed a Hopfield neural network. The network has 2n states, the value of each neuron is 1 or 0 and the output is a four-bit binary number. There are 16 network states in total due to the network's recursive characteristics. If the network is stable, it converges from any initial state to a stable state. If the network is unstable, it will not result in a divergent state since each neuron has only two states.

Jordan [15] proposed a learning structure in 1986 that included labeling the sequence signal with an output that feedback to hidden nodes. The system multiplies the input X value by the now weight value in the neural network, adds the deviation value in the node and recalculates the value to provide the output value. This architecture achieves better results for an RNN modeled by serial signals where the before and after signals are correlated. Therefore, the architecture is often applied to natural language, handwriting recognition, weather, or sensor-detected values. However, as the hidden layer of the network becomes deeper, vanishing or exploding gradient problems may occur, rendering the architecture unsuitable in situations where the signal time is long.

2.3 Long Short-Term Memory (LSTM)

The LSTM is a recurrent neural network. Since the RNN architecture is unsuitable for very long time series, Hochreiter et al. [16] have proposed the LSTM architecture. The architecture has 8 input nodes, 4 output nodes and 2 memory blocks of size 2.

The LSTM architecture consists of three layers, the forget gate, the input gate and new input, and the output gate. The forget gate determines which old memories to retain, the input gate and the new input determine which values need to be added and the output gate determines which values to output.

2.4 Activation Function

The activation function in neural networks mainly uses nonlinear equations to solve nonlinear problems. When the activation function is not used, the neural network is a linear combination of operations. As both the hidden layer and the output layer are the input results of the upper layer, they are calculated with the weight value and the deviation, and the calculation result is regarded as the output of the layer so that the output and the input have a linear relationship. If a nonlinear activation function is not used, the learned model cannot solve a nonlinear problem. In this work, we have investigated 4 different activation functions for learning the framework and selected suitable functions through experiments to implement the fall detection model.

2.4.1 Rectified Linear Unit (ReLU)

According to Eq. (1), when the value of x is negative, the output is 0, and when x is positive, the output remains unchanged. Since the algorithm is linear, the result is easy to predict. The quick convergence provides effective solutions for vanishing and exploding gradient problems with very few calculations. However, when a certain neuron is 0, it will be difficult to activate this node and the node will have no effective results for the data.

$$ReLU\,(x) = \begin{cases} x, & x > 0 \\ 0, & x \le 0 \end{cases} \tag{1}$$

2.4.2 Exponential Linear Unit (ELU)

According to Eq. (2), when the value of x is negative, a nonlinear function is used to provide the output value. This function resolves the shortcomings of the ReLU with an increased amount of calculations.

$$ELU\,(x) = \begin{cases} x, & x > 0 \\ \sigma\,(e^x - 1), & x \le 0 \end{cases} \tag{2}$$

2.4.3 Hyperbolic Tangent Function

The function is calculated according to Eq. (3) and it compresses the output values to the range of between −1 and 1. The effect is better when the feature is obvious. However, the function requires more complex exponential calculations and the gradient vanishing problem still occurs when the input value is extremely large or small.

$$tanh\,(x) = \frac{\boldsymbol{e}^{\boldsymbol{x}} - \boldsymbol{e}^{-x}}{\boldsymbol{e}^{\boldsymbol{x}} + \boldsymbol{e}^{-x}} \tag{3}$$

2.4.4 Sigmoid Function

The function is provided in Eq. (4), which compresses the output value to the range of between 0 and 1. The effect is better with the bi-partition. Nevertheless, the function requires exponential calculations and the gradient vanishing problem may still occur.

$$Sigmoid\,(x) = \frac{1}{1 + \boldsymbol{e}^{-x}} \tag{4}$$

3 System Structure

This research incorporates a three-axis gyroscope, a three-axis acceleration sensor, as well as the Bluetooth radio frequency technology, Raspberry Pi system, management terminal interface program, cloud database and other related components and technology to implement the smart fall detection system. The system also uses artificial intelligence algorithms to improve the accuracy of fall detection.

In this work, we propose an efficient solution for the monitoring center to confirm the status of a user wearing the device via a convenient and intuitive operation management platform. The system detects accidental falls using a three-axis gyroscope and a three-axis accelerometer. The Bluetooth wireless transmitter is used to send the current user status to the signal receiving device. A packet is then sent through the TCP/IP Socket to the computer interface program for status recording and display. The system needs to distinguish between various falling postures and other actions performed by the user, as well as overcome electromagnetic interference caused by metallic or electronic objects in the experimental area to increase the accuracy and reliability of fall detection.

3.1 Hardware Architecture

The system architecture is shown in Fig. 1. The architecture of the system includes four hardware components: the devices worn by the user, information receivers for receiving packets, routers for establishing the local networks and computers with interface programs.

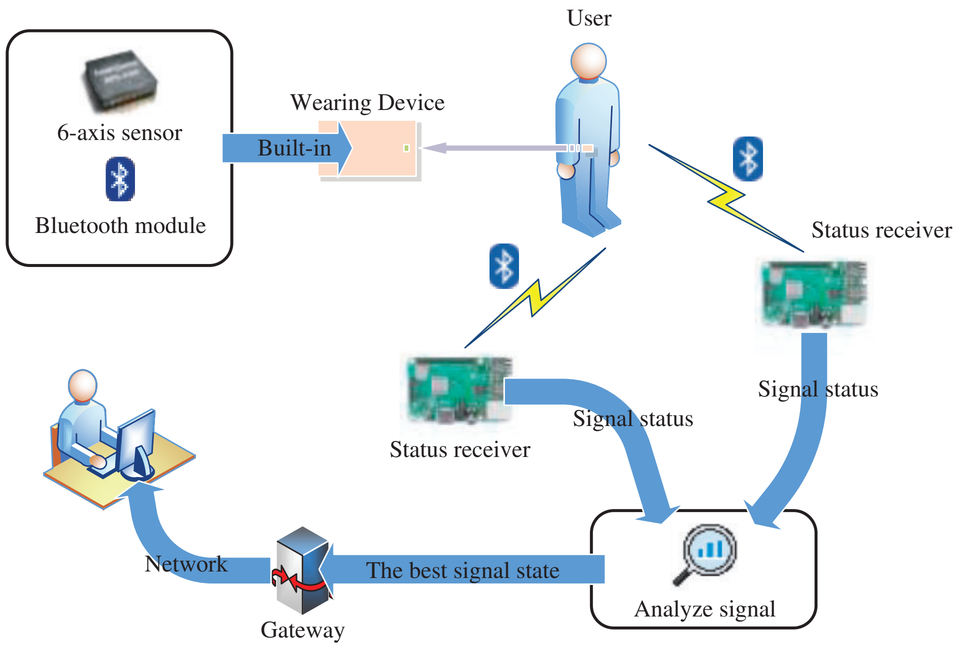

Figure 1: The architecture of the smart fall detection system

As the device worn by the user obtains the six-axis signal, the smart fall detection algorithm is used to calculate the current walking state of the user and information packets are continuously sent to the information receiving device through the Bluetooth broadcast mode. The information receiver calculates the RSSI value of the received packet and selects the information receiver with the best RSSI value for data transmission. When the information receiver receives the relevant signal, it will be sent to the management computer through the wired network. The signal will be analyzed through the smart fall detection model to find the current walking status of the user and the user status will be displayed on the interface program. The management computer will also store the analysis data on the cloud database.

The wearing device uses the Texas Instruments CC2541 Bluetooth wireless chip. The core of the chip is an 8051 microcontroller and a six-axis sensing module (MPU-6050). The fall detection algorithm analyzes the signals generated during the user's actions to determine if a fall has occurred. The power source of the device is a 3.7 V, 1000 mAh, polymer lithium battery and its working endurance is about 30 days.

3.2 Smart Fall Detection System Process

When the user attaches the device to the waist and turns the power on, the power level is automatically detected, which allows the management staff to confirm if the power of the device is sufficient and check the status of the wearing device. The smart fall detection model on the wearing device starts to continuously detect for a fall event and uses the information packet to send the detected state to the information receiver for transmission. The information receiver analyzes whether or not the received packet needs to be forwarded to the management computer and discard the packets sent by the non-wearing device.

The status information of the correct packet is sent to the management computer through the TCP/IP Socket, the content of the information packet is analyzed through the computer interface program and the result is displayed on the graphical user interface. The management personnel is able to check the status information of all users in the field through the computer interface program. When a person accidentally falls or activates the emergency button, the displayed user marker will be marked in different colors and an alarm page will be displayed, allowing the management staff to quickly proceed to the location and deal with the situation. Fig. 2 shows the flowchart of the fall detection system.

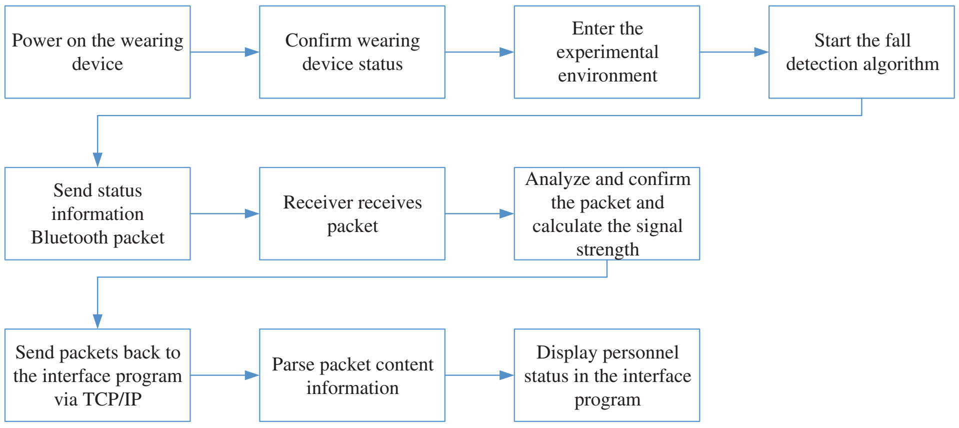
Power on the wearing device
Confirm wearing device status
Enter the experimental environment
Start the fall detection algorithm
Send status information Bluetooth packet
Receiver receives packet
Analyze and confirm the packet and calculate the signal strength
Send packets back to the interface program via TCP/IP
Parse packet content information
Display personnel status in the interface program

Figure 2: Flowchart of the fall detection system

3.3 *Smart Fall Detection Model Establishment*

To solve the difference between the fall and non-fall states of a user, we use a neural network to build a detection model and find an optimal algorithm to solve the problem. The development software for building the model is Spyder and the model is implemented using the Python programming language and Tensorflow framework.

The proposed system uses a variety of neural network models to find a model that consumes fewer resources and is highly accurate. The established model is programmed in the C/C++ language to reproduce the algorithm and apply it to smart fall detection. Fig. 3 shows the flowchart of the smart fall detection model.

3.3.1 Acquire Training Materials

To capture the complete fall action, all 6 signals from the six axes are used and every 80 times is taken as a complete action. The once time is 10 ms. A total of 110 fall actions and 130 other actions are captured as the training material.

3.3.2 Build Model Architecture

In this work, 3 different neural network models have been chosen for comparison with respect to the accurate determination of the differences between each action. The three models are the multi-layer perceptron (MLP), recurrent neural network (RNN) and long short-term memory

(LSTM) models. When building the model, 160 motion signals are used for training, and 80 signals are used for testing the trained model. The input layer is 6 * 80 groups of fall signal values and the output layer is 2 to indicate if a fall action has been detected. After modifying the number of nodes, the number of layers, the activation function of the intermediate hidden layer, and considering the resource constraints of the controller, it is impossible to store too many model weights. The system uses the Tensorflow framework to train a model that meets the demands and select a model with low resource consumption and high accuracy out of the three chosen models.

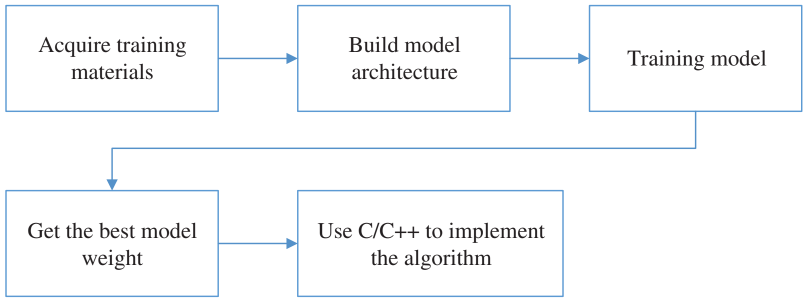
Acquire training materials
Build model architecture
Training model
Get the best model weight
Use C/C++ to implement the algorithm

Figure 3: Flowchart of the smart fall detection model

3.3.3 Multi-Layer Perceptron Fall Model

The architecture diagram of the MLP fall model is shown in Fig. 4. The MLP is a fully connected layer architecture. It inputs 480 action signal values simultaneously. Each node needs to be multiplied and added to all of the input values and the point with the largest value along with the output node is taken as the result. For example, the largest output in the first output node means that a fall action has been detected and the largest output in the second node means that a non-fall action has been detected.

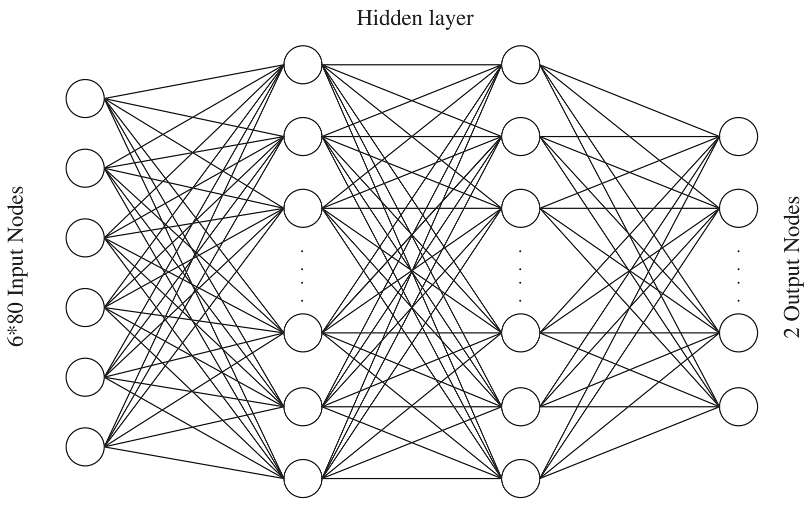

Figure 4: Architecture diagram of the MLP fall model

3.3.4 Recurrent Neural Network Fall Model

The architecture diagram of the RNN fall model is shown in Fig. 5. The input layer of the RNN is different from the MLP. It has only 6 instead of 480 values. Since the RNN is a recursive architecture, only one six-axis signal is given as the input each time (6 signals constitute one input). After completing the calculation, the next six-axis signal is used as the input and the calculation is combined with the previous result. After 80 consecutive inputs, the model uses the calculation result of the fully connected layer to determine whether or not there is a fall event. This model is better at finding the correlation of the input data as a continuous signal and the effect of the RNN training is better than that of the MLP. Therefore, when training the model, another output result is not only to detect a fall but other six actions including walking, sitting, squatting and bending over.

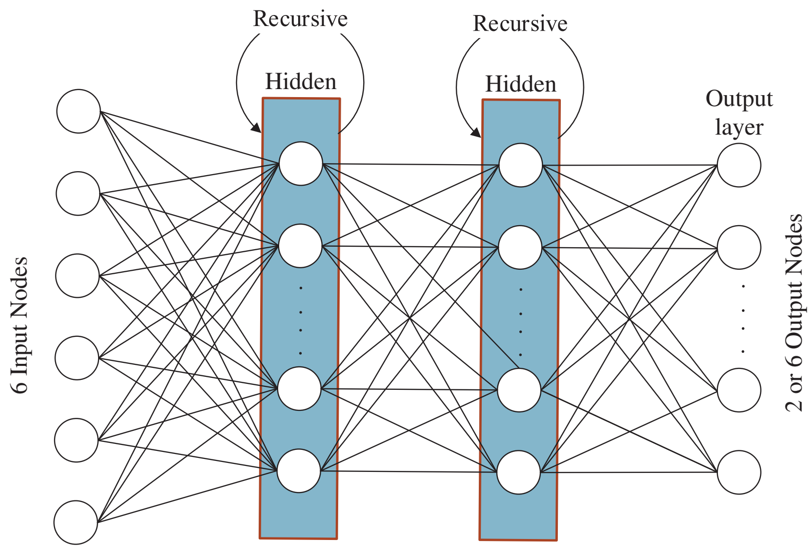

Figure 5: Architecture diagram of the RNN fall model

3.3.5 Long Short-Term Memory Fall Model

The architecture diagram of the LSTM fall model is shown in Fig. 6. The LSTM is an improved version of the RNN architecture. The RNN uses the last six-axis signal to predict the current state, whereas the LSTM not only remembers the previous result but also the result of the previous calculation and uses the forget gate to delete unnecessary information. If the continuous signal of the input data is related to a previous signal as well as another earlier signal, then the LSTM can be used to learn a more effective model and detect six actions like the RNN model.

4 Experimental Result and Discussion

Corresponding correction methods and discussions are proposed according to the test data recorded in the smart fall detection model from training and testing. The proposed system is evaluated with respect to the model test data collection, model architecture accuracy and resource consumption.

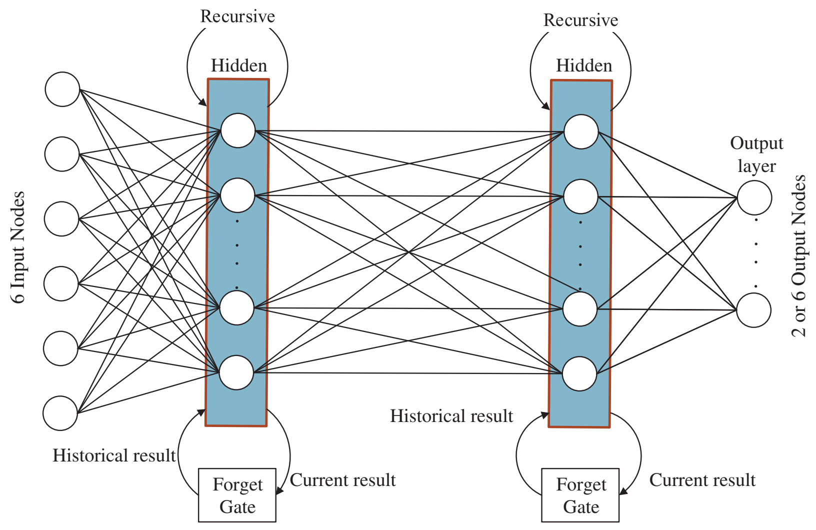

Figure 6: Architecture diagram of the LSTM fall model

4.1 Wearable Device Signal Stability

The system adopts a non-fixed fall detection system. We install the wearable device on the test person. The accelerometer sensor on the wearable device will detect the falling state of the test person at any time and determine whether a fall has occurred. Since the acceleration stability is different when acting in different positions. This experiment will install the wearable device on the chest, wrist and waist to test the stability of the impact.

In Fig. 7, the stability of the wearable device installed on the chest and waist is regarded as the best. However, when it is installed on the chest, wearing it for a long time will cause discomfort and inconvenient installation. Therefore, the wearable device of the system is installed at the waist position and designed belt-type.

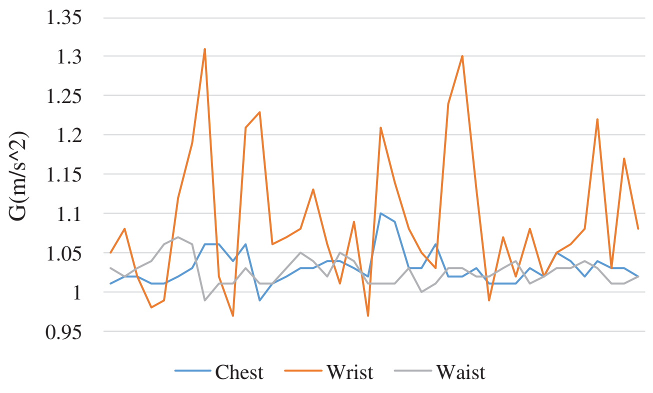

Figure 7: Signals for wearable device

4.2 Model Test Data Acquisition

To capture the complete fall action, the 6 signals of the six axes are acquired once every 10 ms and every 80 acquisitions constitute a complete action, as shown in Figs. 8a–8e. In Fig. 8, the horizontal axis represents the number of times and the vertical axis represents the signal values from the six axes in different colours. The red box in the figure is the six-axis signal captured during the action. To avoid classifying non-fall actions as a fall, a total of 5 other actions were also selected for training, so that the model would not misjudgments other actions. A total of 240 action signals were captured, 110 of which fall actions and 130 were other, non-fall actions.

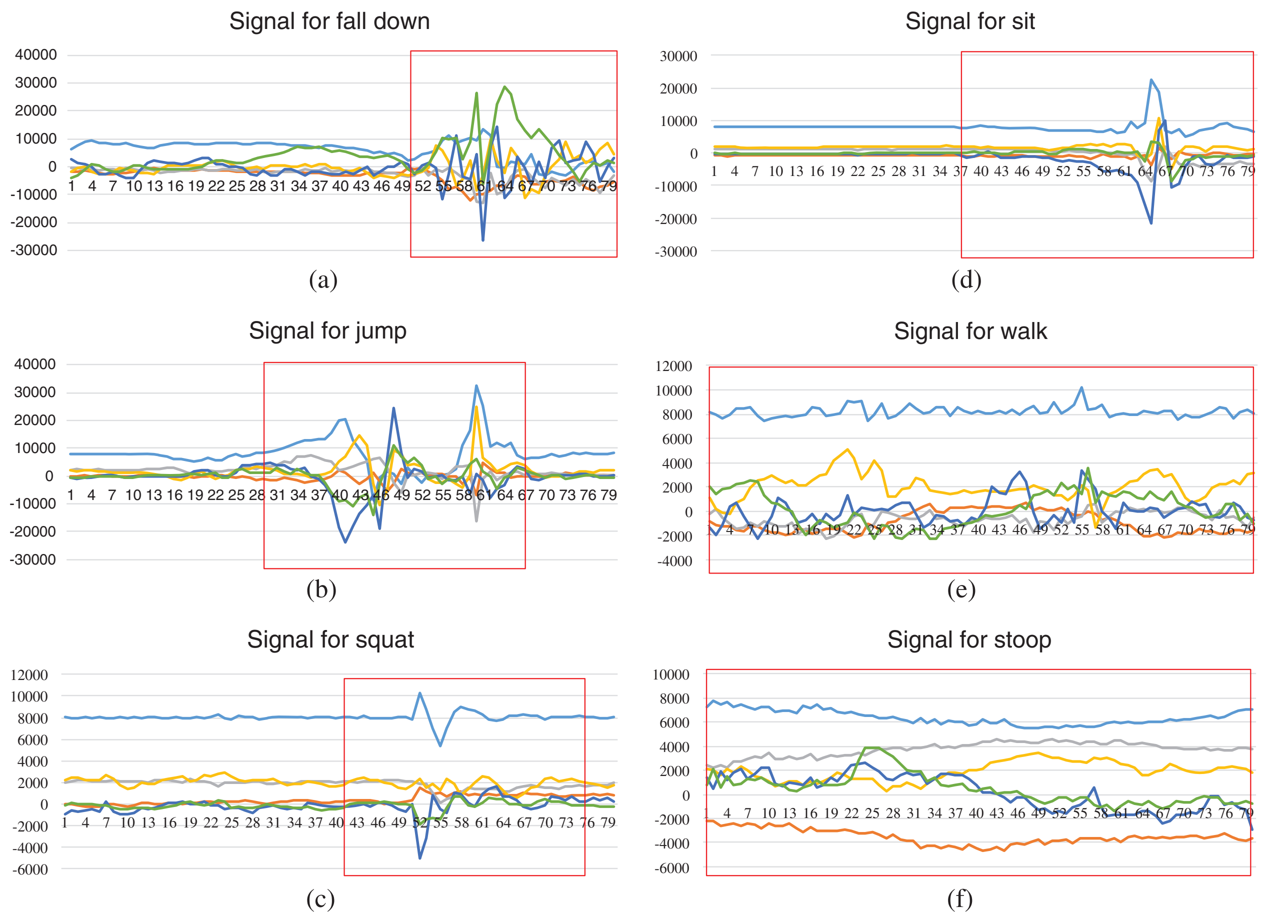

Figure 8: Six-axis signals for six actions

4.3 *Model Architecture Accuracy and Resource Consumption*

The six-axis signal is captured every 10 ms and if the weight value is too large, it cannot be burned into the microcontroller. Therefore, the calculation time of the model must be within 10 ms and the resources consumed by the number of node weights must also be met. The learning effect varies according to the hidden layer, the number of nodes and the activation function. In this work, we have selected several model architectures with good results in training and testing. The final test model does not show a large difference between the training and testing accuracies, which means that the model has high reliability.

4.3.1 MLP Fall Model Accuracy and Resource Consumption

Since each layer of the MLP architecture is a fully connected layer with 480 input nodes, where each node in the first layer has 480 weight values and 1 deviation. As a result, the number of weight values is very large and the microcontroller cannot load the resources. In the end, the algorithm is not implemented in the fall detection model and the resource consumption is not calculated. Tab. 2 is the model architecture for the MLP training and the activation function uses the ReLU to reduce the number of model calculations.

Table 2: The model architecture for MLP Training

Judge action	Number of layers	Number of nodes	Dropout	Activation function
A fall action is detected	1 layer	6	0.7	ReLU
A fall action is detected	1 layer	9	0.7	ReLU
A fall action is detected	2 layers	1st layer 4 2nd layer 6	0.7	ReLU
A fall action is detected	2 layers	1st layer 6 2nd layer 9	0.7	ReLU

Tab. 3 shows the results of the MLP model training. It can be seen from Tab. 3 that the accuracy increases as the number of nodes or layers increases. This is due to the training data having too many features and using few nodes for training, resulting in poor performance. Conversely, when the number of nodes is sufficient with respect to the number of features, the comparison result is improved. Although the calculation time meets the requirements, the weight of the architecture far exceeds the resources of the microcontroller, therefore, this model cannot be implemented on the wearable device.

Table 3: The results for MLP model training

Judge action	Number of weights	Training accuracy (%)	Testing accuracy (%)	Calculated time (ms)
A fall action is detected	2886 hidden layers 14 output layers	88	86	4.1
A fall action is detected	4329 hidden layers 20 output layers	92	90	5.6
A fall action is detected	1954 hidden layers 14 output layers	93	92	3.3
A fall action is detected	2949 hidden layers 20 output layers	95	94	4.3

4.3.2 RNN Fall Model Accuracy and Resource Consumption

Since the trained RNN model has better results, the model is divided into two architectures to judge a fall action and other six kinds of actions, to facilitate better understandings of the user's current actions and prevent a user from performing dangerous actions, such as a jump. If a user's action before the fall can be detected, the rescue can be carried out more effectively.

The number of weights on a single node in the hidden layer of the RNN is the number of input nodes plus the number of nodes in the hidden layer. For example, if the input value of the hidden layer is 6 and the number of hidden nodes is 12, then each node of the layer has 18 weight values and 1 deviation value. Tab. 4 is the model architecture for RNN training. To adapt the implementation for the microcontroller, fewer nodes are used.

Table 4: The model architecture for RNN training

Judge action	Number of layers	Number of nodes	Dropout	Activation function
A fall action is detected	1 layer	6	0.7	ReLU
A fall action is detected	1 layer	9	0.7	ReLU
Six actions	2 layers	1st layer 12 2nd layer 18	0.7	ReLU
Six actions	2 layers	1st layer 18 2nd layer 24	0.7	ReLU

Tab. 5 shows the results of the RNN model training. From Tab. 5, it can also be seen that when the number of nodes increases, the accuracy also increases because when the number of nodes increases, the number of historical features that can be remembered also increases and improves the accuracy. However, it can be seen in this table that adding nodes to the six-action model increases the accuracy gap between training and testing, which means that the model is slightly over-fitting. Since too many nodes have been used for training, the model only satisfies the training data.

Table 5: The results for RNN model training

Judge action	Total number of nodes	Training accuracy (%)	Testing accuracy (%)
A fall action is detected	6 hidden nodes 2 output nodes	96	95
A fall action is detected	9 hidden nodes; 2 output nodes	99	98
Six actions	1st layer 12 hidden nodes 2nd layer 18 hidden nodes 6 output nodes	96	95
Six actions	1st layer 18 hidden nodes 2nd layer 24 hidden nodes 6 output nodes	97	93

Tab. 6 is the model resource for the RNN implementation for fall detection. Since the implemented model is an architecture for fall detection, only the time of a fall event is calculated in this model and the number of weights consumed by the model is calculated to ensure that it can be implemented on the microcontroller.

4.3.3 LSTM Multiple-Action Fall Model Accuracy and Resource Consumption

Since the trained LSTM model has better results, it has two architectures like the RNN model. Each layer of the LSTM model has 4 sets of nodes to determine the memory and output values. The number of hidden layer nodes will be 4 times the setting and the weight value of each node is the same as the RNN, which is the input plus the recursion number. For example, if the hidden layer is set to 12 nodes, the input value is 6, which means that the layer has 48 nodes, and each node has 18 weight values and a deviation value. Tab. 7 is the model architecture for the LSTM training.

Table 6: The model resource for the RNN implementation for fall detection

Architecture	Number of weights	One-time calculation time (ms)
1-layer RNN hidden layer with 6 nodes 1-layer fully connected output layer with 2 output nodes	78 hidden layers 14 output layers	4.8
1-layer RNN hidden layer with 9 nodes 1-layer fully connected output layer with 2 output nodes	144 hidden layers; 20 output layers	5.8

Table 7: The model architecture for the LSTM training

Judge action	Number of layers	Number of nodes	Dropout	Activation function
A fall action is detected	1 layer	6	0.7	sigmoid tanh
A fall action is detected	1 layer	9	0.7	sigmoid tanh
Six actions	2 layers	1^{st} layer 9 2^{nd} layer 12	0.7	sigmoid tanh
Six actions	2 layers	1^{st} layer 12 2^{nd} layer 18	0.7	sigmoid tanh

Tab. 8 is the result for LSTM model training. It can be seen from the result that the model is slightly overfitting with regard to the fall detection model because the number of nodes explained above will be 4 times the set, which is too many for training. In the six-action model, each action has its own different characteristics, so more nodes are required for learning and the final result is not over-fitting.

Tab. 9 is the six-action model resources implemented by LSTM. Since the implemented model is used to judge six actions, only the time for the six actions taken is calculated in this model. It can be seen from the table that the calculation time of this model far exceeds the time taken to grab a six-axis signal and the weight value also far exceeds the resources of the microcontroller, so this model cannot be implemented on the wearable device.

4.3.4 The Fall Model Comparison

Tab. 10 shows the comparison between the three chosen architectures. Only the settings with the highest accuracy are selected for comparison for each architecture. Although the accuracy of the MLP fall architecture is not as good as the other two, it is the fastest of the three in terms of calculation time, and although it consumes fewer resources than the LSTM, the microcontroller is still unable to load. The RNN fall architecture has the highest accuracy among the three and requires very few resources. However, it is slower than the MLP architecture due to its recursive nature, but it still meets the demand. Although the LSTM fall architecture is not good at judging a fall, it has good performance for the six actions, but with the worst resource consumption among the three architectures. According to the comparison result, the RNN fall architecture is selected for the proposed system.

Table 8: The results for the LSTM model training

Judge action	Total number of nodes	Training accuracy (%)	Testing accuracy (%)
A fall action is detected	24 (4 times 6) hidden nodes 2 output nodes	97	94
A fall action is detected	36 (4 times 9) hidden nodes 2 output nodes	98	94
Six actions	1st layer 36 hidden nodes 2nd layer 48 hidden nodes 6 output nodes	95	94

Six actions	1st layer 48 hidden nodes;
	2nd layer 72 hidden nodes;
	6 output nodes
		97	95

Table 9: The six-action model resources implemented by LSTM

Architecture	Number of weights	One-time calculation time (ms)
2-layers LSTM hidden layer with 84 nodes 1-layer fully connected output layer with 6 output nodes	1632 hidden layers 78 output layers	20
2-layers RNN hidden layer with 120 nodes 1-layer fully connected output layer with 6 output nodes	3144 hidden layers 114 output layers	30

Table 10: The comparison between the three chosen architectures

Comparison items	Model	Number of nodes	Number of weights	Time consumption (ms)	Accuracy rate (%)
MLP fall architecture	A fall action is detected	1st layer 6 hidden nodes; 2nd layer 9 hidden nodes; 2 output nodes	2969	4.3	94.5
RNN fall architecture	A fall action is detected	9 hidden nodes; 2 output nodes	164	5.8	98.5
LSTM fall architecture	Six actions	1st layer 18 hidden nodes 2nd layer 24 hidden nodes 6 output nodes	3258	30	96

5 Conclusion

In this work, we have proposed a smart fall detection system that uses artificial intelligence to implement fall detection to improve the accuracy of human fall detection. The proposed system uses a variety of different neural network models to learn a low-resource and high-accuracy algorithm. The system provides the users with a convenient management interface program so that the management staff can quickly and efficiently handle accidental situations. The experimental results have demonstrated that the accuracy of fall detection can be up to 98%.

At present, artificial intelligence is widely used to facilitate systems that require large amounts of data and noise processing. Still, the model architecture often requires too many resources to be written into the microcontroller, such as the LSTM algorithm used in this research. In the future, if the processing resources of the hardware microcontroller become sufficient, artificial intelligence can be effectively applied to various signal-related systems.

Funding Statement: The authors received no specific funding for this study.

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

References

1. Lapierre, N., Neubauer, N., Miguel-Cruz, A., Rincon, A. R., Liu, L. et al. (2018). The state of knowledge on technologies and their use for fall detection: A scoping review. *International Journal of Medical Informatics, 111(6),* 58–71. DOI 10.1016/j.ijmedinf.2017.12.015.
2. Ren, L., Peng, Y. (2019). Research of fall detection and fall prevention technologies: A systematic review. *IEEE Access, 7,* 77702–77722. DOI 10.1109/ACCESS.2019.2922708.
3. Gates, S., Fisher, J. D., Cooke, M. W., Carter, Y. H., Lamb, S. E. (2008). Multifactorial assessment and targeted intervention for preventing falls and injuries among older people in community

and emergency care settings: systematic review and meta-analysis. *BMJ, 336(7636),* 130–133. DOI 10.1136/bmj.39412.525243.BE.

4. Faes, M. C., Reelick, M. F., Joosten-Weyn Banningh, L. W., Gier, M. D., Esselink, R. A. et al. (2010). Qualitative study on the impact of falling in frail older persons and family caregivers: Foundations for an intervention to prevent falls. *Aging & Mental Health, 14(7),* 834–842. DOI 10.1080/13607861003781825.
5. Wang, Y., Wu, K., Ni, L. M. (2016). Wifall: Device-free fall detection by wireless networks. *IEEE Transactions on Mobile Computing, 16(2),* 581–594. DOI 10.1109/TMC.2016.2557792.
6. Yin, X., Shen, W., Samarabandu, J., Wang, X. (2015). Human activity detection based on multiple smart phone sensors and machine learning algorithms. *IEEE 19th International Conference on Computer Supported Cooperative Work in Design*, pp. 582–587. Calabria, Italy.
7. Chelli, A., Pätzold, M. (2019). A machine learning approach for fall detection and daily living activity recognition. *IEEE Access, 7,* 38670–38687. DOI 10.1109/ACCESS.2019.2906693.
8. De Miguel, K., Brunete, A., Hernando, M., Gambao, E. (2017). Home camera-based fall detection system for the elderly. *Sensors, 17(12),* 2864. DOI 10.3390/s17122864.
9. Lu, N., Wu, Y., Feng, L., Song, J. (2018). Deep learning for fall detection: Three-dimensional CNN combined with LSTM on video kinematic data. *IEEE Journal of Biomedical and Health Informatics, 23(1),* 314–323. DOI 10.1109/JBHI.2018.2808281.
10. Shahzad, A., Kim, K. (2018). FallDroid: An automated smart-phone-based fall detection system using multiple kernel learning. *IEEE Transactions on Industrial Informatics, 15(1),* 35–44. DOI 10.1109/TII.2018.2839749.
11. Ichwana, D., Arief, M., Puteri, N., Ekariani, S. (2018). Movements monitoring and falling detection systems for transient ischemic attack patients using accelerometer based on Internet of Things. *International Conference on Information Technology Systems and Innovation*, pp. 491–496. Bandung, Indonesia.
12. Nooruddin, S., Islam, M. M., Sharna, F. A. (2020). An IoT based device-type invariant fall detection system. *Internet of Things, 9(Suppl 2),* 100130. DOI 10.1016/j.iot.2019.100130.
13. Clemente, J., Li, F., Valero, M., Song, W. (2019). Smart seismic sensing for indoor fall detection, location, and notification. *IEEE Journal of Biomedical and Health Informatics, 24(2),* 524–532. DOI 10.1109/JBHI.2019.2907498.
14. Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America, 79(8),* 2554–2558. DOI 10.1073/pnas.79.8.2554.
15. Jordan, M. I. (1986). *Serial order: A parallel distributed processing approach (No. 8604 ICS Technical Report)*. San Diego, La Jolla, CA: University of California.
16. Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9(8),* 1735–1780. DOI 10.1162/neco.1997.9.8.1735.