DIGITAL HEALTH
Volume 10: 1–15
© The Author(s) 2024
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/20552076241259047
journals.sagepub.com/home/dhj

TCS-Fall: Cross-individual fall detection system based on channel state information and time-continuous stack method

Ziyu Zhou[1], Zhaoqing Liu[1], Yujie Liu[1], Yan Zhao[1], Jiarui Wang[1], Bowen Zhang[2], Youbing Xia[1], Xiao Zhang[1] and Shuyan Li[1]

Abstract

Background: Falls pose a serious health risk for the elderly, particular for those who are living alone. The utilization of WiFi-based fall detection, employing Channel State Information (CSI), emerges as a promising solution due to its non-intrusive nature and privacy preservation. Despite these advantages, the challenge lies in optimizing cross-individual performance for CSI-based methods.

Objective: This study aimed to develop a resilient real-time fall detection system across individuals utilizing CSI, named TCS-Fall. This method was designed to offer continuous monitoring of activities over an extended timeframe, ensuring accurate and prompt detection of falls.

Methods: Extensive CSI data on 1800 falls and 2400 daily activities was collected from 20 volunteers. The grouped coefficient of variation of CSI amplitudes were utilized as input features. These features capture signal fluctuations and are input to a convolutional neural network classifier. Cross-individual performance was extensively evaluated using various train/test participant splits. Additionally, a user-friendly CSI data collection and detection tool was developed using PyQT. To achieve real-time performance, data parsing and pre-processing computations were optimized using Numba's just-in-time compilation.

Results: The proposed TCS-Fall method achieved excellent performance in cross-individual fall detection. On the test set, AUC reached 0.999, no error warning ratio score reached 0. 955 and correct warning ratio score reached of 0.975 when trained with data from only two volunteers. Performance can be further improved to 1.00 when 10 volunteers were included in training data. The optimized data parsing/pre-processing achieved over 20× speedup compared to previous method. The PyQT tool parsed and detected the fall within 100 ms.

Conclusions: TCS-Fall method enables excellent real-time cross-individual fall detection utilizing WiFi CSI, promising swift alerts and timely assistance to elderly. Additionally, the optimized data processing led to a significant speedup. These results highlight the potential of our approach in enhancing real-time fall detection systems.

Keywords

Channel state information, cross-individual fall detection, grouped coefficient of variation, real-time detection, time-continuous stack sample

Submission date: 21 December 2023; Acceptance date: 15 May 2024

[1]School of Medical Information and Engineering, Xuzhou Medical University, Xuzhou, China
[2]School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China

Corresponding authors:
Xiao Zhang, School of Medical Information and Engineering, Xuzhou Medical University, Tongshan Road, Yunlong District, Xuzhou, Jiangsu 221004, China.
Email: changshui@hotmail.com

Shuyan Li, School of Medical Information and Engineering, Xuzhou Medical University, Tongshan Road, Yunlong District, Xuzhou, Jiangsu 221004, China.
Email: lishuyan@xzhmu.edu.cn

Introduction

Background

Falls are the leading cause of fatal and nonfatal injuries among older people, especially those living alone. Society faces significant healthcare issues caused by an aging population. The number of people aged over 65 is expected to reach 1.6 billion by 2050.[1] The World Health Organization (WHO) reported that about 35% of them fall at least once a year.[2] Approximately 20 million older adults in China fall at least once annually. The direct medical cost of these falls is approximately 5 billion yuan, while the indirect social costs reach about 60–80 billion yuan.[3] Approximately 25% of adults aged 65 and over in the United States have experienced a fall, resulting in an annual expenditure of $30 billion in medical costs.[4,5] The mortality rate from falls has been observed to increase, especially in the elderly over the age of 75.[6,7] Effective and prompt fall detection strategies may profoundly benefit society. In other words, the earlier the falls are detected, the lower the mortality rate.[8] Quick and timely detection of these falls is crucial to providing urgent assistance.

Method of fall detection

The following three approaches can be used to perform fall detection: wearable device-based, vision-based and ambiance device-based.[9] Wearable devices must be fixed to the human body at a certain position,[10] which can cause a disconnection problem. This makes wearable devices unfavorable, especially for the elderly. Continuous surveillance using vision-based systems can yield satisfactory detection results.[11] However, this approach raises ethical concerns related to confidentiality and privacy, issues that are currently under active consideration and resolution. Using ambiance devices, such as radar, infrared array sensors, microphone array sensors or Wi-Fi devices, can help circumvent the aforementioned issues. For instance, radar-based methods may detect fall motions by extracting features from radar-reflected signals using deep neural network models.[12] Infrared array sensor-based methods can detect fall motions based on temperature changes between consecutive frames.[13] Microphone array sensor system comprises a circular microphone array that captures sounds in a room. When a sound is detected, the system locates the source, enhances the signal and classifies it as “fall” or “non-fall.”[14] The Wi-Fi-based fall detection method relies on channel state information (CSI), which contains multi-path information from the transmitter to the receiver. This method detects falls by analyzing the interference of human activity on amplitude and phase information of CSI. Among these, WiFi offers advantages such as contactless, confidentially and widespread use.

Channel state information

In a typical indoor environment, Wi-Fi signals propagate through multiple paths via ceilings, floors, walls and furniture, resulting in “multipath effects.” Wi-Fi commonly employs orthogonal frequency division multiplexing (OFDM) modulation that divides high-speed data into multiple low-speed data streams modulated on different orthogonal subcarriers. In multiple-input multiple-output (MIMO) systems, the receiver can obtain CSI corresponding to multiple transmit antennas. CSI reflects the amplitude attenuation and phase shift characteristics of the multiple propagation paths from the transmitter to the receiver. When a human body moves in the Wi-Fi environment, its scattering affects the original propagation paths of the wireless signals. The received signals will contain information about human motion in the environment. By analyzing CSI across OFDM subcarriers and multiple antenna pairs in MIMO systems, applications in lip language recognition,[15] people counting,[16] user authentication,[17] gesture recognition,[18,19] moving speed estimation,[20] activity recognition[21–24] and fall detection[25–35] can be achieved.

CSI-based fall detection

By analyzing the amplitude and phase information of CSI across OFDM subcarriers and multiple antennas in MIMO systems, substantial progress has been made in CSI-based fall detection. The fine-grained multipath profile contained in CSI can capture minute motions and environmental changes caused by human falls. Various machine learning techniques like random forests (RF),[25] support vector machine (SVM),[26–31] convolutional neural network (CNN)[32,33] and long short-term memory (LSTM)[30,34,36] have been applied on these features to build robust fall detection models. Despite the tremendous effort, most existing methods do not distinguish between individuals during model training and testing. This approach achieves high accuracy for the target user but results in poor generalization for new individuals.

For instance, Shalaby et al.[32] collected CSI from six volunteers, but they amalgamated all the CSI data before cross-validation. Meanwhile, Guo et al.[21] gathered CSI from 10 volunteers and attained an accuracy of 93.50% within the same individual dataset. Nevertheless, when applied to a cross-individual dataset, the accuracy dropped to approximately 60%. To overcome this challenge, cross-individual models that can maintain high performance across different users are needed.

Aims of the study

The aim of this study was to develop the TCS-Fall method for cross-individual real-time fall detection using CSI. This method was designed to offer continuous monitoring of activities over an extended timeframe, ensuring accurate and prompt detection of falls. Additionally, a user-friendly tool for the collection and analysis of CSI data was developed to facilitate ease of use and enhance the overall applicability of the proposed fall detection system.

Methods

Ethics approval

The data collection was approved by the Medical Ethics Committee of the Affiliated Hospital of Xuzhou Medical University (Document No.: AF-35/06.1, Opinion No.: XYFY2021-KL269-01). Written informed consent was obtained from all study participants prior to the commencement of the study, as required by the Ethics Committee.

Study design

This research is a method development and evaluation study, which involves the collection of a substantial amount of WiFi CSI data from 20 volunteers, the creation of a deep learning model capable of performing real-time cross-individual fall detection, and the development of a corresponding application tool. The study was carried out between July and August 2022 in the home simulation laboratory at Xuzhou Medical University.

In the data collection part, WiFi CSI was meticulously gathered from a diverse group of 20 volunteers (equally divided between male and female). This encompassed data from 1800 falls, 2400 confusing events and 36,000 s of time-continuous actions.

In the model construction part, the novel TCS-Fall method was introduced, utilizing the grouped coefficient of variation (GCV) of CSI amplitudes as input features for a CNN classifier. Cross-individual performance was thoroughly assessed with various train/test volunteer splits, and model performance was evaluated on test data. In addition to the commonly used AUC metric, this study employed the no error warning ratio (NEWR) and correct warning ratio (CWR) metrics to assess the performance of time-continuous fall detection.

In the tool development part, a data collection, data preprocessing and fall detection tool was developed based on the PyQT package in a Python environment. To achieve real-time performance, data parsing and preprocessing computations were optimized using Numba's just-in-time (JIT) compilation.

Data collection (study 1)

Room environment. The data were collected in the room with one transmitter (Tx), one signal receiver (Rx), two chairs, a mattress ($180 \times 200 \times 25$ cm), a table with a book, a waterglass and a mobile phone, as shown in Figure 1. All the volunteers performed daily activities and fell around chairs A and B.

Hardware and software. A computer with an Intel Core i3-3240 CPU was used as the transmitter, and a computer with an Intel Core i5-9400F CPU was employed as the receiver. Both computers were equipped with an IEEE 802.11n Intel Wi-Fi Link 5300 chipset. The CSI tool proposed by Halperin et al.[37] was deployed on computers to collect CSI data.

Volunteer's attributes. Twenty volunteers, comprising of 10 males and 10 females, were recruited to participate in this experiment in order to mitigate any potential gender-related errors. Their ages ranged from 19 to 26 years old, heights from 153 to 181 cm, and weights from 40 to 80 kg to control for differences in body composition. The detailed data are shown in Table 1.

Single-action and time-continuous stack samples. A series of single-action (SA) samples and time-continuous stack (TCS) samples were collected from 20 volunteers. The actions detailed in Table 2 were repeated multiple times from different angles, directions and postures. This approach ensured a comprehensive and varied dataset for analysis.

For SA samples, 8 s of data were collected in each sample, repeated 30 times. SA samples were collected from 1800 falls and 2400 confusing events. For TCS samples, 150 s of data were collected in each sample, repeated two times. In the initial 120 s, volunteers engaged in various daily activities. In the final 30 s, volunteers experienced a single fall at a randomly chosen moment, and the time of the fall was recorded. TCS samples were collected from approximately 36,000 s of time-continuous actions.

Data format. All filenames were saved in the format of "a_[Volunteer ID]_[Action ID]_[Repetition times].dat." The volunteer ID ranged from 1 to 20, representing the 20 volunteers in the data collection. The Action ID refers to the activity being performed, as detailed in Table 2. The activities include falls as well as daily routines like sitting, standing, walking, etc. Repetition times indicate the repeat number of that action, ranging from 0 to 29.

For example, the filename "a_01_02_03.dat" indicates the data file for volunteer with ID "01" performing action "02" for the fourth time, since the repetition is "03." This systematic naming format allowed clear organization of the extensive dataset collected in the study.

Model construction (study 2)

Get CSI data from file with numba. This study analyzed the construction of binary ".dat" files based on the initial

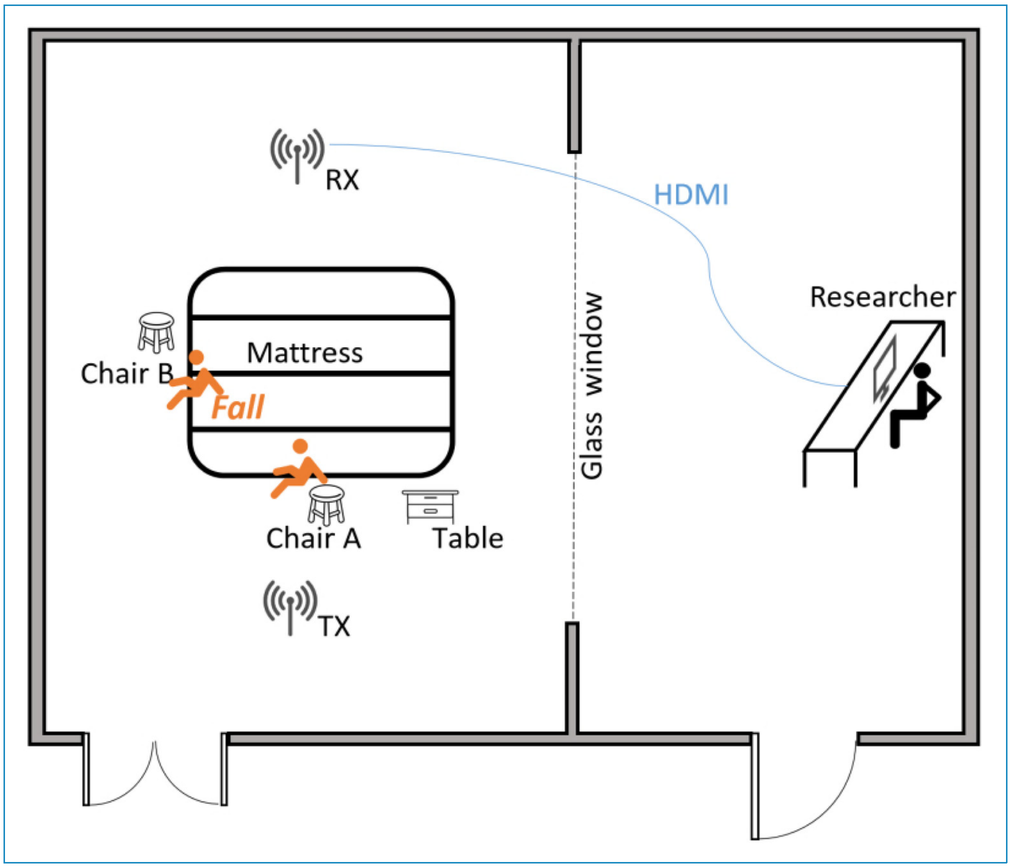


Figure 1. Room environments.

version of csitool released by Halperin et al.[37,38] The binary file contains multiple packets. The binary file contains multiple packets, and the structure of each packet is illustrated in Figure 2. Each block in the figure represents one byte.

When the value of "code" is 187, the packet contains CSI information; otherwise, it contains other information. In the packet, the length of the packet is parsed using big-endian byte order, while the other parameters are parsed using little-endian byte order. The "Ntx" field means the number of transmit antennas, while the "Nrx" field means the number of receive antennas. The fields "rssi_a," "rssi_b" and "rssi_c," respectively, represent the received signal strength indication (RSSI) on antennas A, B and C. The "noise" field represents the level of background noise present in the received signal. The field "agc" represents the automatic gain control value, indicating the level of signal amplification adjustment performed by the receiver. The "antenna" field denotes the antenna selection, indicating which antennas are being utilized for reception. The "fake_rate_n_flags" field denotes the fake rate and associated flags, providing information about the simulated transmission rate and additional flags.

The CSI data is stored in the field "payload," the structure of which is illustrated in Figure 3. For the sake of illustration, the figure depicts two subcarriers, each containing only two data points. Each small block in the image represents 1 bit, and 8 blocks make up one byte.

To improve the speed, the method for parsing CSI from Payload was optimized using Numba, as presented in Algorithm 1. In this algorithm, the "mod" function refers to the modulo operation, the "div" function refers to the division operation and the "int8" function is used to convert a numerical value to an 8-bit signed integer type.

Algorithm 1 Parse CSI from Payload with Numba

```
Input: payload: The bytes data contained csi.
Input: Ntx, Nrx: Number of transmit/receive antennas.
Output: CSI: The complex-valued CSI array.
CSI ← $\mathbb{C}^{Ntx \times Nrx \times 30}$
index ← 0
For i ← 1 to 30 do
    index ← index + 3
    r ← mod(index, 8)
    For j ← 1 to Nrx do
        For k ← 1 to Ntx do
            s ← div(index, 8)
            real ← (payload[s] >> r)|(payload[s + 1] << (8 − r))&255
```

Table 1. Volunteer's attributes.

Volunteer ID	Gender	Age	Height	Weight
1	Male	23	173	80
2	Female	20	161	52
3	Female	19	167	68
4	Female	20	168	54
5	Female	19	162	45
6	Male	24	168	62
7	Female	24	153	40
8	Female	22	153	43
9	Female	21	167	56
10	Male	21	180	70
11	Male	23	178	69
12	Female	26	160	45
13	Male	25	166	75
14	Male	26	168	63
15	Male	26	181	70
16	Male	24	174	58
17	Female	23	165	48
18	Female	22	162	78
19	Male	24	176	73
20	Male	23	160	55

```
imag ← (payload[s + 1] >> r)|(payload[s + 2] << (8 − r)) &255
CSI[k, j, i] ←int8(real) + int8(imag) × 1j
index ← index + 16
End for
End for
End for
Return CSI
```

To eliminate the effects of attenuation and interference as much as possible,[38] the csi data must be scaled as illustrated in Figure 4. The "dbinv()" function in the figure converts dBm values to linear power values, and the "GetTotalRss()" function calculates the total RSSI from a CSI struct.

Table 2. Detail of SA samples and TCS samples.

Activities	Activity ID	Repeat	Seconds	With Fall?
SA samples				
Fall	0	30	8	With Fall
Fall while walking	1	30	8	With Fall
Fall while standing up	2	30	8	With Fall
Sit on a chair	3	30	8	Without Fall
Stand up from a chair	4	30	8	Without Fall
Squat	5	30	8	Without Fall
Stand up from Squat	6	30	8	Without Fall
TCS samples				
Keep standing	10	2	150	In final 30 s
Keep standing with daily activities	11	2	150	In final 30 s
Keep Walking	12	2	150	In final 30 s
Walk with daily activities	13	2	150	In final 30 s
Keep sitting on a chair	14	2	150	In final 30 s
Keep sitting with daily activities	15	2	150	In final 30 s

Standardization of samples. The SA and TCS samples data were standardized in the same format. For the SA samples, each sample contained $8 \times 1000 = 8000$ packets. For the TCS samples, an 8-s time window was adopted to slide forward by 1 s each time to obtain the sliced item as shown in Figure 5. Each sliced item contained $8 \times 1000 = 8000$ packets.

The communication between the transmitter and receiver employs 3×3 MIMO data. Each transmitter–receiver pair

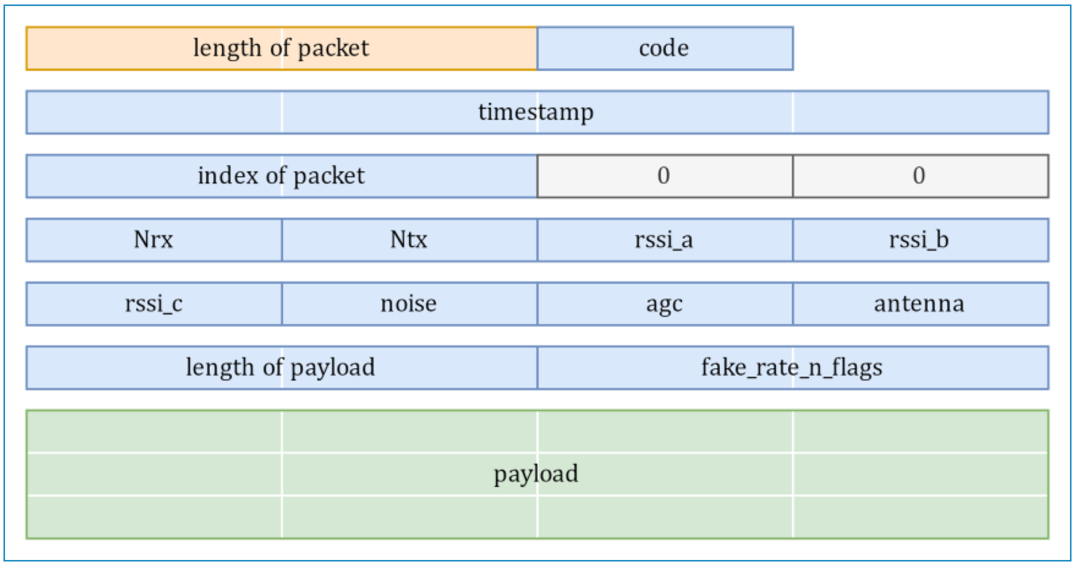


Figure 2. Structure of the binary packet.

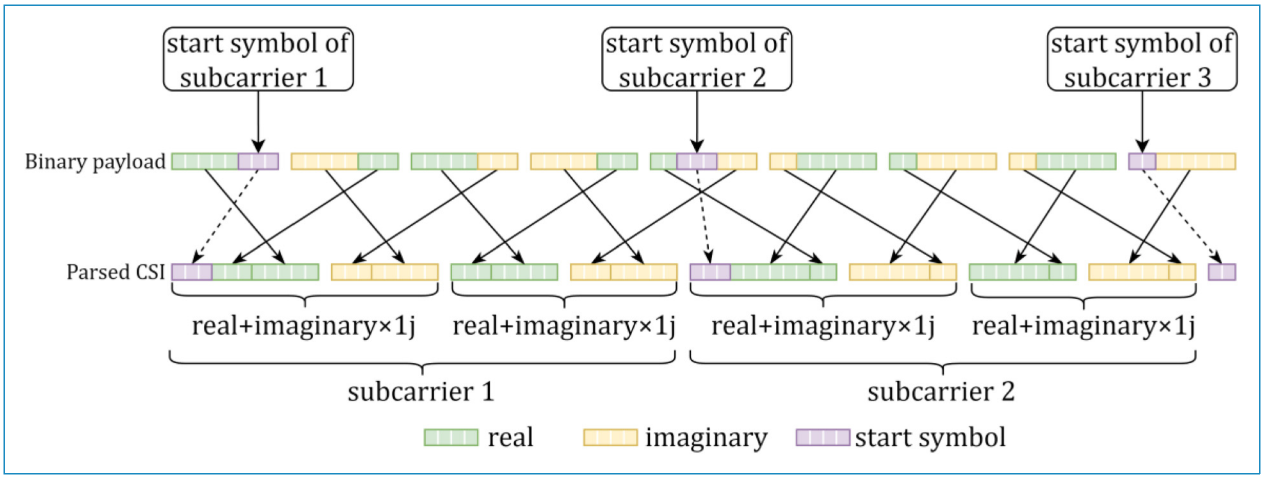


Figure 3. Structure of the binary payload.

comprises 30 groups of subcarriers reported by the Intel 5300 NIC.[37] A standardized sample includes $8000 \times 3 \times 3 \times 30$ elements. Due to the homogeneity of the transmitter data, only $8000 \times 1 \times 3 \times 30 = 8000 \times 90$ elements were used in this study. The original amplitude signals for various actions are shown in Figure 6(a)–(c).

Dataset split. Training and test dataset: As shown in Figure 5, the training and test dataset comprised SA samples and sliced items extracted from the initial 120 s of TCS samples. SA samples containing falls were designated as positive samples, while the rest were labeled as negative samples. To avoid the over-fitting problem and perform cross-individual identification, the k-fold method was employed to divide the training and test sets based on the volunteer ID rather than the sample ID. If a volunteer is assigned to the training set, all the samples generated by that volunteer will also be allocated to the training set.

This study employed three modes to divide the training and test sets to evaluate the model's performance under different training data sizes. Mode 1: 10-fold cross-validation is used, with data from 2 volunteers used for training and data from the other 18 volunteers used for testing in each validation. Mode 2: 2-fold cross-validation is used, with data from 10 volunteers used for training and data from the other 10 volunteers used for testing in each validation. Mode 3: 10-fold cross-validation is used, with data from 18 volunteers used for training and data from the other 2

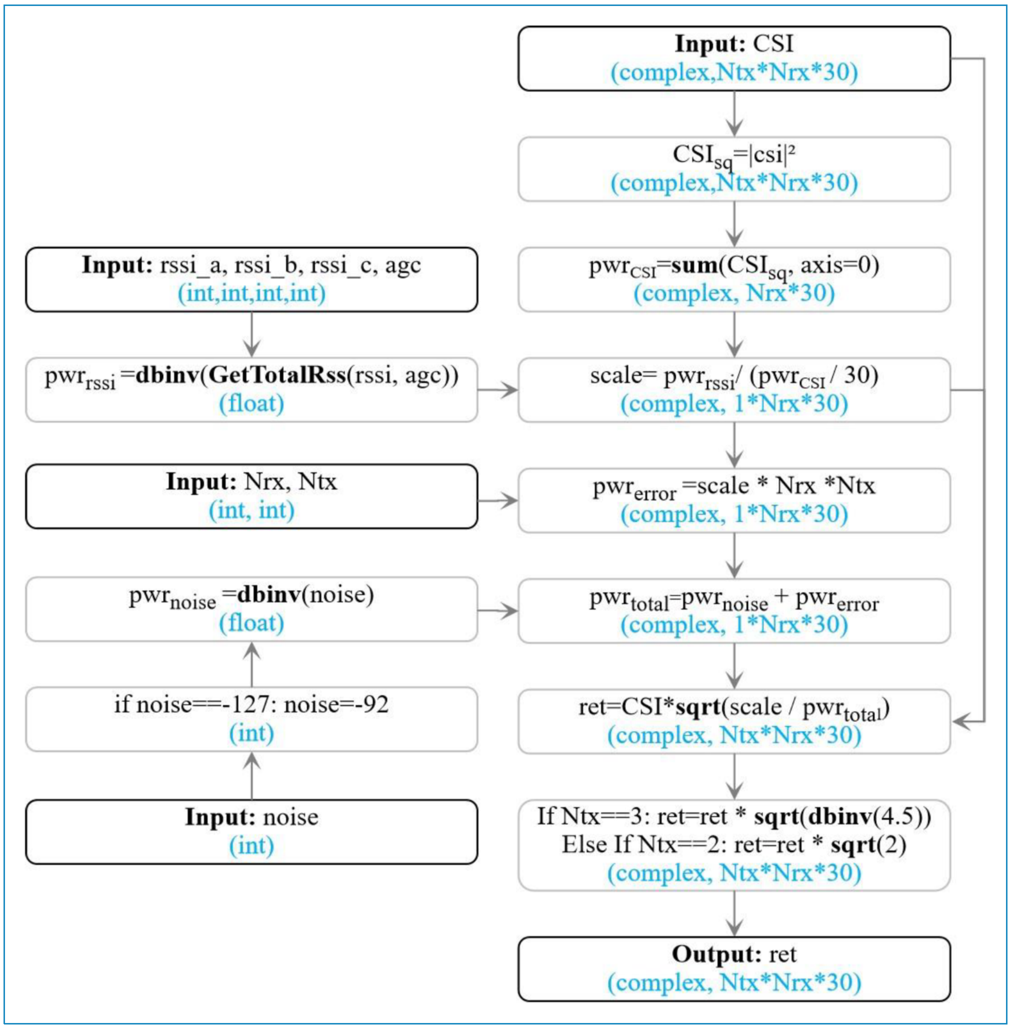

Input: CSI
(complex,Ntx*Nrx*30)
CSI_sq=|csi|²
(complex,Ntx*Nrx*30)
pwr_CSI=sum(CSI_sq, axis=0)
(complex, Nrx*30)
scale= pwr_rssi/ (pwr_CSI / 30)
(complex, 1*Nrx*30)
pwr_error =scale * Nrx *Ntx
(complex, 1*Nrx*30)
pwr_total=pwr_noise + pwr_error
(complex, 1*Nrx*30)
ret=CSI*sqrt(scale / pwr_total)
(complex, Ntx*Nrx*30)
If Ntx==3: ret=ret * sqrt(dbinv(4.5))
Else If Ntx==2: ret=ret * sqrt(2)
(complex, Ntx*Nrx*30)
Output: ret
(complex, Ntx*Nrx*30)
Input: rssi_a, rssi_b, rssi_c, agc
(int,int,int,int)
pwr_rssi =dbinv(GetTotalRss(rssi, agc))
(float)
Input: Nrx, Ntx
(int, int)
pwr_noise =dbinv(noise)
(float)
if noise==-127: noise=-92
(int)
Input: noise
(int)

Figure 4. Get scaled CSI with numba.

volunteers used for testing in each validation. To mitigate the effects of randomness, all evaluation metrics presented in the results are calculated as the average of k-fold cross-validation.

TCS dataset: As shown in Figure 5, the TCS dataset consisted of all sliced items extracted from TCS samples. This dataset was utilized to assess the model's performance over an extended duration. To avoid data overlap, only the data from volunteers in the test dataset was used during the model evaluation.

Noise filter. Butterworth filter is the most commonly used infinite impulse response (IIR) filters.[28] It does not introduce any ripple in the passband or stopband, making it ideal for applications where a smooth frequency response is required.

In this step, the bidirectional Butterworth low-pass filter was used to remove the background noise in the experimental environment with high frequency above 100 Hz. The original and filtered amplitude signals of different actions are shown in Figure 6 (d)–(f).

GCV of CSI amplitude. To extract features of CSI amplitude variations, this article employs the GCV method. GCV can reflect the degree of change in CSI amplitude and is not influenced by baseline values. The calculation formula is as follows:

$$f_{GCV}(i,j) = f_{CV}(X_{iw:(i+1)w,j}),\ i \in \{1,2,3,\ldots,\lfloor \frac{8000}{w} \rfloor\},\ j \in \{1,2,3,\ldots,90\}. \quad (1)$$

Here, i is the time-axis index, j is the channel index of CSI, w is the window-size of each group and X represents the CSI matrix. $X_{iw:(i+1)w,j}$ represents the sub-matrix of X consisting of the i-th window from $i \times w$ to $(i+1) \times w$ and carrier j. The function $f_{CV}(x)$ represents the coefficient of variation for the w values within the current window, and

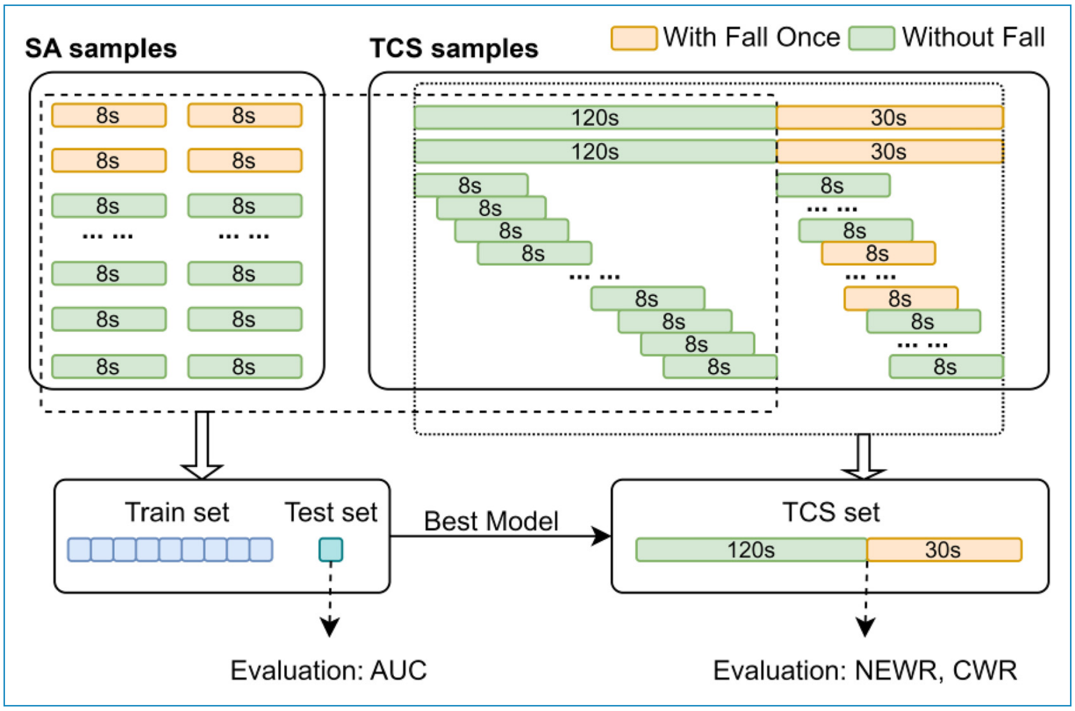


Figure 5. Standardization of samples and dataset split.

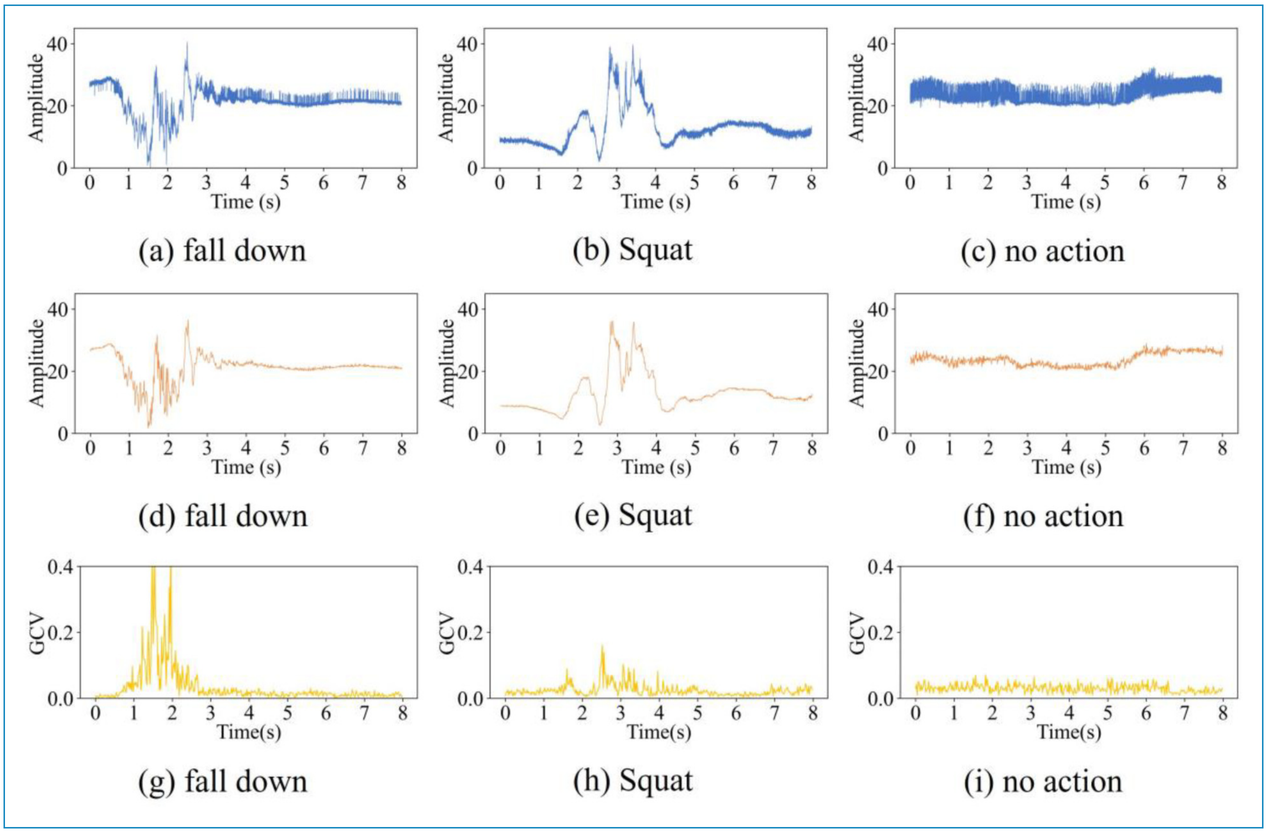


Figure 6. The waveforms of different actions. (a–c): the original amplitude signal of CSI. (d–f): the filtered amplitude signal of CSI. (g–i): the GCV features of CSI.

Table 3. The CNN model of TCS-fall.

Layer	Filter size and neurons	Pooling operation	Activation function	Additional operations
Input layer	1 400×90	None	None	None
1st Convolution	64 5×5 filters	2×2 Max Pooling	ReLU	0.6 Dropout
2nd Convolution	128 5×5 filters	5×3 Max Pooling	ReLU	0.6 Dropout
3rd Convolution	128 5×5 filters	5×3 Max Pooling	ReLU	0.6 Dropout
4th Convolution	128 5×5 filters	2×1 Max Pooling	ReLU	0.6 Dropout
5th Convolution	64 5×5 filters	None	ReLU	0.6 Dropout
Fully connected	None	None	ReLU	None
Output layer	None	None	SoftMax	None

its calculation formula is:

$$f_{CV}(x) = \frac{\sqrt{\frac{\sum_{i=1}^{w}(x_i - \bar{x})^2}{w}}}{\bar{x}}. \tag{2}$$

The w adjacent amplitudes in each carrier wave were grouped into a window and their GCV was calculated. By this way, multiple carrier waves were transformed into a new matrix. For example, if there are Ntx × Nrx × 30 carrier waves and each carrier wave contains 8000 amplitudes, then a (8000/w × Ntx × Nrx × 30) matrix will be obtained after this transformation.

After dimensional reduction by GCV, the dimension of CSI signal in time axis is decreased to 1/w of the original signal. These GCV features are fed as inputs into the neural network model, which can effectively improve the classification performance. By adjusting the size of w, data compression and computational efficiency can be balanced. Excessive compression may lead to reduced features while insufficient compression cannot effectively optimize efficiency. In this study, w = 20 is chosen through optimization to achieve the best balance between efficiency and feature integrity. Finally, a $400\times1\times3\times30$ matrix is obtained and used as inputs for the subsequent neural network process. The GCV features of different actions are shown in Figure 6(g)-(i).

Standardization of data. To obtain better performance of the neural network models, each sub-carrier data was converted into a matrix with a mean of 0 and a standard deviation of 1, as shown in the following formula:

$$X_{standard} = \frac{X - \mu}{\sigma}. \tag{3}$$

The symbol μ represents the mean of X, and σ represents the standard deviation of X.

Convolution neural network model. Firstly, we evaluated three to six-layer CNN networks and found out that a five-layer CNN effectively minimized computation time while preserving a high accuracy rate. As shown in Table 3, the final best CNN model comprises an input layer, five convolutional layers, a fully connected layer and an output layer.

The neural network comprises an input layer that receives a 400×90 CSI data matrix. Five convolutional layers follow, each consistently applying a dropout rate of 0.6 after convolution and max-pooling operations to prevent over-fitting. ReLU activation is employed in all convolutional layers, introducing nonlinearity. The first convolutional layer uses 64 filters of size 5×5 with 2×2 max pooling. Subsequently, the second and third convolutional layers utilize 128 filters of size 5×5 with 5×3 max pooling. The fourth convolutional layer employs 128 filters of size 5×5 with 2×1 max pooling. The fifth convolutional layer applies 64 filters of size 5×5 without pooling.

The extracted features undergo flattening and are fed into a fully connected layer, which uses ReLU activation. Finally, the output layer, employing SoftMax activation, classifies the features for accurate fall detection. The consistent application of regularization and activation functions in each convolutional layer contributes to robust feature learning and the non-linear transformation of CSI data.

Evaluation metrics. Evaluation on test datasets: The metrics used for the evaluation of the test dataset were area under the curve (AUC) score. AUC is a statistical measure that evaluates the performance of a binary classification model by measuring the area under the receiver operating characteristic (ROC) curve and can easily calculated with the python package "scikit-learn."[40]

Evaluation on TCS dataset: Each sliced item of TCS sample corresponds to an 8-s activity. It is easy to know if each sliced item contains a fall by comparing the fall time for TCS sample. A TCS sample was recorded as error warnings if any sliced item without a fall was erroneously predicted as a fall. In contrast, a TCS sample was recorded as correct warnings if any sliced item with a fall was correctly predicted as a fall. The evaluation metrics used for the TCS dataset are NEWR and CWR. These metrics used to evaluate the time-continuous fall detection effect of the model are calculated as shown in the following formulas:

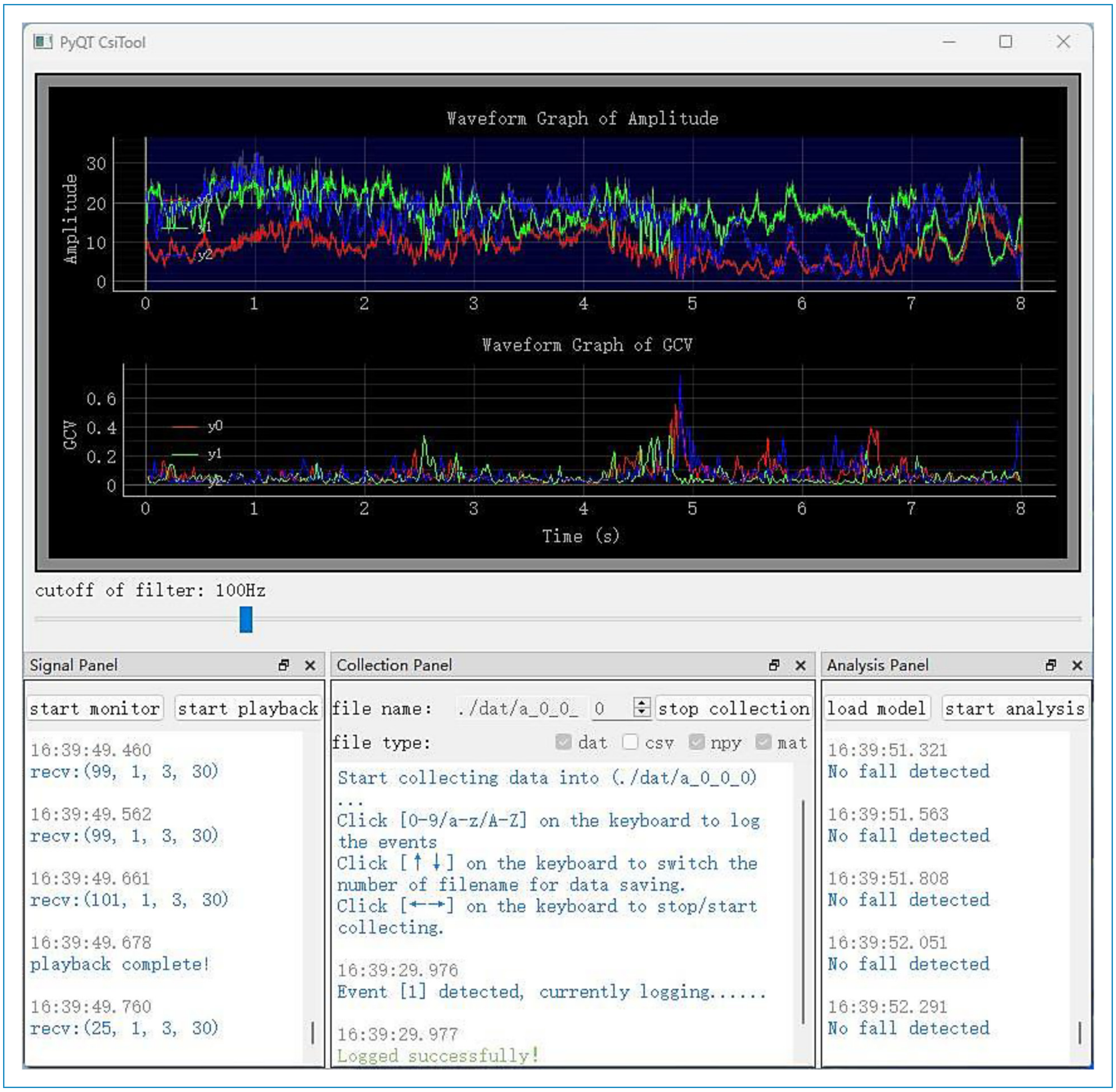
$$NEWR = 1 - \frac{num\ of\ \ error\ warnings}{num\ of\ \ TCS\ samples}, \tag{4}$$

$$CWR = \frac{num\ of\ \ correct\ warnings}{num\ of\ \ TCS\ samples}. \tag{5}$$

Tool development (study 3)

Main window of PyQT CsiTool. A program called "PyQT CsiTool" to facilitate CSI data collection and human activity detection was developed here in this work. This program can perform functions such as signal reception, CSI data collection, event marking, and human activity detection.

The program's main window contains four panels: waveform display area, signal panel, collection panel and analysis panel as shown in Figure 7.



```
cutoff of filter: 100Hz
```

```
16:39:49.460
recv:(99, 1, 3, 30)

16:39:49.562
recv:(99, 1, 3, 30)

16:39:49.661
recv:(101, 1, 3, 30)

16:39:49.678
playback complete!

16:39:49.760
recv:(25, 1, 3, 30)
```

```
file name:  ./dat/a_0_0_  0
file type:        dat  csv  npy  mat
Start collecting data into (./dat/a_0_0_0)
...
Click [0-9/a-z/A-Z] on the keyboard to log
the events
Click [↑ ↓] on the keyboard to switch the
number of filename for data saving.
Click [←→] on the keyboard to stop/start
collecting.

16:39:29.976
Event [1] detected, currently logging......

16:39:29.977
Logged successfully!
```

```
16:39:51.321
No fall detected

16:39:51.563
No fall detected

16:39:51.808
No fall detected

16:39:52.051
No fall detected

16:39:52.291
No fall detected
```

Figure 7. PyQT CsiTool.

Waveform display area. The primary function of waveform display area is to display the signal waveform, including the waveform graph of amplitude, the waveform graph of GCV and slider of a cutoff frequency slider.

The original and filtered amplitude are displayed in dark and bright color tones, respectively. To modify the Butterworth low-pass filter's cutoff frequency, users only need to drag the slider. The waveform graph of GCV displays the GCV values relative to the amplitude.

Signal panel. The primary function of the signal panel is to select the signal source. CSI signal extraction from the Intel 5300 network card can only be done in the Ubuntu operation system, while the playback function can be used in both Windows and Ubuntu.

Collection panel. The primary function of the collection panel is to save the CSI signal and event data. The operator only needs to input the name prefix and select the saved file type such as "dat," "csv," "npy" and "mat." Then, the program will automatically increase the file name number after each data collection. To record an event, press [0–9, a-z, A-Z] on the keyboard, then the event and its occurrence time will be recorded in the CSV file with the same name.

Analysis panel. The analysis panel depends on the open neural network exchange (ONNX)[41] file exported from the neural network model. After importing the fall detection model, the panel will continuously transmit the latest amplitude data to the classification model and display the prediction results.

In addition to the fall detection model mentioned in this article, any neural network models such as action recognition, individual recognition, gesture recognition, person counting, etc., can also be loaded as long as they were built using the ONNX framework.

Results

Data collection (study 1)

The dataset encompasses 4200 SA samples, capturing falls and actions closely resembling falls. Additionally, it contains 18,000 s of TCS samples, featuring daily actions with falls inserted.

Performance of model (study 2)

Performance of proposed method. A k-fold cross-validation method was employed to enhance the reliability of the experimental results. The evaluation findings are shown in Table 4. This method demonstrated robustness. The proposed method yielded satisfactory results even with the utilization of CSI data from only two volunteers for training. It is worth highlighting that the model's proficiency increases as the size of the training dataset grows.

Table 4. Evaluate performance using varying numbers of volunteers for training.

Mode	AUC	NEWR	CWR
Mode 1[a]	0.999	0.955	0.975
Mode 2[b]	0.999	1.000	0.992
Mode 3[c]	0.999	1.000	1.000

[a]Mode 1: 10-fold cross-validation is used, with data from 2 volunteers used for training and data from the other 18 volunteers used for testing in each validation.
[b]Mode 2: 2-fold cross-validation is used, with data from 10 volunteers used for training and data from the other 10 volunteers used for testing in each validation.
[c]Mode 3: 10-fold cross-validation is used, with data from 18 volunteers used for training and data from the other 2 volunteers used for testing in each validation.

Performance comparison of different filter parameters. In order to obtain the optimal model performance, different filtering methods were tested on the CSI data and the results were compared.

First, models were trained using Butterworth low-pass filter, Hampel filter, Gaussian filter and without filtering. The results showed that only the Butterworth low-pass filter could significantly improve the model performance. Other filtering methods may filter out important fall information and reduce performance (see Table 5).

Then, bidirectional Butterworth low-pass filters with different cutoff frequencies were tested. High cutoff frequencies retained more signal components but also more noise, while low cutoff frequencies filtered out more noise but potentially useful signal components as well. As shown in Table 5, using a 100 Hz cutoff achieved the optimal balance—removing noise while retaining useful information for fall detection.

In summary, using the Butterworth low-pass filter with a 100 Hz cutoff frequency to preprocess the CSI data was chosen, in order to obtain the optimal model performance.

Optimization of window-size for GCV. To extract features of CSI amplitude variations, the GCV method is utilized in this study. As shown in Algorithm 1, the window-size w determines the degree of data compression by GCV. Table 5 demonstrates our exploration with different values of w (i.e. 5, 10, 20 and 40). From the results, it is evident that a larger w leads to higher data compression but may discard useful features for detecting falls, reducing the CWR. In contrast, a smaller w retains more features but cannot effectively optimize computational efficiency.

Table 5. The influence of different parameters on model performance[a].

Filter method	Cut-off frequency	Window-size	AUC	NEWR	CWR
None	–	20	0.997	0.806	0.997
Gaussian	–	20	0.999	0.981	0.912
Hampel	–	20	0.995	0.797	0.99
Butterworth	50 Hz	20	0.999	0.981	0.945
Butterworth	100 Hz	20	0.999	0.955	0.975
Butterworth	150 Hz	20	0.999	0.883	0.991
Butterworth	100 Hz	5	0.995	0.85	0.966
Butterworth	100 Hz	10	0.998	0.907	0.988
Butterworth	100 Hz	40	0.999	0.957	0.952

[a]This performance is from 10-fold cross-validation, where data from 2 volunteers was used for training and data from the other 18 volunteers was used for testing in each validation.

To achieve the best balance between retaining useful features and improving efficiency, we selected $w = 20$ through optimization. This window-size achieves high NEWR and CWR scores simultaneously. In summary, adjusting the window-size allows balancing between data compression and preserving features critical for fall detection.

Performance comparison with other models. Based on our current knowledge, the majority of existing fall detection methods utilizing WiFi CSI rely on non-cross-individual mixed data for training and testing purposes, without reporting cross-individual fall detection results. Therefore, we compared several recently published works with our proposed TCS-Fall method using this cross-individual dataset. As shown in Table 6, the utilization of GCV features in our proposed method offers a significant advantage in reducing NEWR compared to other methods in fall detection.

Table 6. Predictive performance across different methods[a].

Features and models	AUC	NEWR	CWR
SVD and RF [25]	0.95	0.615	0.939
PCA + STFT and SVM [27]	0.97	0.652	0.944
Amplitude and CNN [32]	0.981	0.521	0.969
PCA + STFT and CNN [33]	0.992	0.893	0.95
GCV and RNN	0.979	0.386	0.975
GCV and LSTM	0.990	0.748	0.966
GCV and CNN (proposed method)	0.999	0.955	0.975

[a]This performance is from 10-fold cross-validation, where data from 2 volunteers was used for training and data from the other 18 volunteers was used for testing in each validation.

Performance of tools (study 3)

Performance of PyQT-CSITool. The PyQT-CSITool program is built on the Python packages "PyQT5" and "Numba." The analysis of CSI binary data employs the JIT precompile technology of Numba to enhance its speed. All operations that have the potential to block threads, including socket listening, data playback, and action recognition, are executed in sub-threads to prevent the main thread from being obstructed.

Through the optimization of multi-threading and precompile technology, the program can operate seamlessly even on a low-performance machine. For example, on a system equipped with an N2840 CPU, this tool can adeptly parse 1 s of CSI data and detects the fall using neural network prediction within 100 ms. With this subsecond latency, the system can detect falls in real-time.

Performance comparison with other tools. While most classification tasks are completed within a negligible timeframe of 5–20 ms,[33] reading and preprocessing CSI data can take several seconds. This significant disparity highlights the need to optimize data pre-processing for real-time applications.

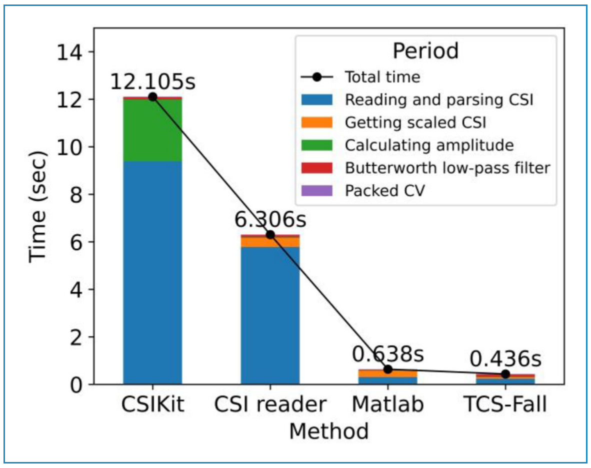


Figure 8. Speed performance of different method.

This study compared the processing times for reading and pre-processing an 8-s CSI data file stored in the ".dat" format. Four methods were evaluated: (a) the Python-based "CSIKit,"[42] (b) the Python-based "CSI reader,"[43] (c) the MATLAB-based method[38] and (d) the proposed method accelerated by Numba. As shown in Figure 8, the proposed method achieved a significant speedup exceeding 20-fold compared to the other approaches.

Discussion

Evaluation metrics NEWR and CWR

Most existing CSI-based fall detection studies use classification metrics like accuracy, precision, recall or AUC to evaluate model performance.[25–35] However, these metrics are suitable for evaluating the classification of independent data samples, but have limitations in assessing continuous fall detection over an extended period.

Specifically, existing metrics cannot effectively evaluate the occurrence of false alarms—when the system erroneously detects falls during daily non-fall activities over a long duration. To address this, we introduced two new evaluation metrics—NEWR and CWR.

NEWR measures the ratio of time windows with no false alarms during normal activities to the total monitoring duration. It indicates how effectively the system can avoid erroneously detecting falls during routine daily activities over time. A higher NEWR means the system has fewer false alarms over time. CWR measures the ratio of time windows that correctly detect falls over the total monitoring duration. A higher CWR ratio means more falls are correctly detected in a timely manner.

Both NEWR and CWR are essential to evaluate real-life performance—the system needs to detect actual falls accurately while avoiding false alarms during normal activities. The extensive experiments in this study demonstrate that the proposed TCS-Fall method can achieve high NEWR and CWR simultaneously, highlighting its ability for continuous cross-individual fall detection.

In this work, if the model was trained using data from only two volunteers, the NEWR was 0.955. This means that given one TCS sample as 150 s long, the probability of no false alarms in this duration is 0.955. Therefore, if a user performed different activities continuously for 1 h (3600 s), the probability of producing a false alarm would be $3600/150 * 0.045 = 1.08$, indicating a high likelihood of approximately 1 false alarm occurrence.

However, when the model was trained using data from 10 volunteers, the NEWR value reached 1. This indicates that the probability of generating false alarms during long periods of continuous motion approaches 0. In such a scenario, no false alarms would be produced throughout an entire day. Therefore, the model trained by 10 volunteers is recommended for the real world applications.

MATLAB or Python&Numba

MATLAB is a common tool used to analyze CSI data because there are many signal processing toolkits available in this software.[21,39] However, it is difficult to perform due to many unwanted functions. It is also difficult to commercialize because of its charging mechanism. These limitations have led some researchers to analyze CSI data using Python, a free, lightweight platform.[29,42]

Python's appeal in data science stems from its efficient, dynamic programming paradigm. However, this very flexibility presents a double-edged sword. While the untyped, high-level syntax simplifies development, it can lead to performance bottlenecks in data- and computation-intensive programs. Compiled code offers significant speed advantages in such scenarios, often running orders of magnitude faster than dynamically interpreted code.[44]

Numba,[45] a JIT compiler, addresses Python's performance limitations by translating Python functions into optimized machine code. By leveraging type information of input parameters, the Numba compiler utilizes the LLVM library to generate efficient machine code, achieving performance comparable to compiled languages like C or Fortran.[46]

By capitalizing on Python's flexibility and Numba's ability to compile code for faster execution, this study achieved a 20-fold performance improvement compared to conventional methods. This finding suggests that the Python-Numba combination offers a compelling alternative to MATLAB, particularly for researchers seeking an open-source, high-performance solution for CSI data analysis.

Limitations

While this paper presents an interesting WiFi-based fall detection system using CSI, there are some limitations that should be considered.

First, the experiments were conducted in a controlled lab environment with volunteers performing scripted actions. Real-world environments can be much more complex and unpredictable.

Second, the demographics of the volunteers were quite homogeneous—mostly young students in their 20 s. Testing in a more diverse population, including the elderly, will be crucial. However, due to the high-risk nature of falls, collecting fall data from elderly individuals is nearly impossible.

Third, the system's hardware limitations necessitate the installation of a dedicated WiFi transmitter and receiver within the home, potentially restricting widespread adoption. In the future, there is a need to develop devices that can seamlessly integrate with existing home WiFi networks to enhance feasibility.

Overall, this research presents promising technical development but further real-world testing across more diverse situations is needed to evaluate limitations and practical viability.

Conclusions

In this study, we proposed TCS-Fall, a novel deep learning method for real-time cross-individual fall detection using WiFi CSI. This method utilizes GCV of CSI amplitudes as input features to effectively capture fluctuations caused by falls. The GCV features are fed into a Convolutional Neural Network classifier optimized through extensive architecture exploration.

The proposed method was evaluated on an extensive dataset collected from 20 volunteers performing falls as well as daily activities. The results demonstrated that TCS-Fall can achieve exceptional cross-individual performance, with AUC reaching 0.999, NEWR score reaching 0.955 and CWR score reaching 0.975 when trained on just 2 volunteers. The accuracy improved further as more training data was utilized.

Additionally, this study developed a user-friendly tool for CSI data collection and analysis using PyQT and optimized the performance using Numba. The optimized data parsing/preprocessing achieved over 20x speedup compared to previous methods. The tool could parse and detect falls within 100 ms, enabling real-time performance.

In conclusion, this study makes significant contributions in addressing the challenge of cross-individual fall detection using WiFi CSI. The proposed TCS-Fall method enables highly accurate real-time detection. Additionally, the optimized data processing and user-friendly tool facilitate practical deployment. This work advances the field and brings us closer to WiFi-based fall monitoring systems that can provide timely assistance to the elderly.

Acknowledgments: We wish to thank Xu Yang and Yuqing Yin in China University of Mining and Technology for their helpful advices. We appreciate ChatGPT's contribution to polish the language of manuscript. It is important to clarify that its role was solely focused on linguistic refinement, without involvement in any content generation, literature creation or knowledge point formulation.

Contributorship: Ziyu Zhou was responsible for the experimental design, programming, data collection, model development and program deployment. Xiao Zhang and Shuyan Li provided guidance and supervision. The other authors assisted with data collection for the experiments.

Data Availability: The data sets generated during or analyzed during this study are available from the corresponding author on reasonable request.

Declaration of Conflicting Interests: The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Ethical approval: The data collection was approved by the Medical Ethics Committee of the Affiliated Hospital of Xuzhou Medical University (Document No.: AF-35/06.1, Opinion No.: XYFY2021-KL269-01).

Funding: The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the national key research and development program of China (2020YFC2006600) and the Unveiling & Leading Project of XZHMU (JBGS202204).

Guarantor: SL

ORCID iDs: Ziyu Zhou https://orcid.org/0009-0005-5394-1941
Shuyan Li https://orcid.org/0000-0001-7028-4166

References

1. Ritchie H and Roser M (2019) Age structure. Available at: https://ourworldindata.org/age-structure (accessed 18 December 2023).
2. Ageing WHO and Unit LC. *WHO global report on falls prevention in older age*. Geneve, Switzerland: World Health Organization, 2008.
3. Wang J, Chen Z and Song Y. Falls in aged people of the Chinese mainland: epidemiology, risk factors and clinical strategies. *Ageing Res Rev* 2010; 9: S13–S17.
4. Bergen G, Stevens MR and Burns ER. Falls and fall injuries among adults aged >= 65 years - United States, 2014. *MMWR-Morbidity and Mortality Weekly Report* 2016; 65: 993–998.
5. Burns ER, Stevens JA and Lee R. The direct costs of fatal and non-fatal falls among older adults - United States. *J Saf Res* 2016; 58: 99–103.
6. Lozano R, Naghavi M, Foreman K, et al. Global and regional mortality from 235 causes of death for 20 age groups in 1990 and 2010: a systematic analysis for the global burden of disease study 2010. *Lancet* 2012; 380: 2095–2128.

7. Hartholt KA, Lee R, Burns ER, et al. Mortality from falls among US adults aged 75 years or older, 2000-2016. *Jama-J of the Am Med Assoc* 2019; 321: 2131–2133.
8. Gurley RJ, Lum N, Sande M, et al. Persons found in their homes helpless or dead. *N Engl J Med* 1996; 334: 1710–1716.
9. Mubashir M, Shao L and Seed L. A survey on fall detection: principles and approaches. *Neurocomputing* 2013; 100: 144–152.
10. Hellec J, Hayotte M, Chorin F, et al. Applying the UTAUT2 model to smart eyeglasses to detect and prevent falls among older adults and examination of associations with fall-related functional physical capacities: survey study. *J Med Internet Res* 2023; 25: e41220.
11. Bayen E, Jacquemot J, Netscher G, et al. Reduction in fall rate in dementia managed care through video incident review: pilot study. *J Med Internet Res* 2017; 19: e339.
12. Maitre J, Bouchard K and Gaboury S. Fall detection with UWB radars and CNN-LSTM architecture. *IEEE J Biomed Health Inform* 2021; 25: 1273–1283.
13. Tao S, Kudo M and Nonaka H. Privacy-preserved behavior analysis and fall detection by an infrared ceiling sensor network. *Sensors (Basel, Switzerland)* 2012; 12: 16920–16936.
14. Li Y, Ho KC and Popescu M. A microphone array system for automatic fall detection. *IEEE Trans Biomed Eng* 2012; 59: 1291–1301.
15. Wang G, Zou Y, Zhou Z, et al. We can hear you with wi-fi!. In: Proceedings of the 20th annual international conference on Mobile computing and networking, 2014, pp.593–604.
16. Tian L, Chen L, Xu Z, et al. A people-counting and speed-estimation system using wi-fi signals. *Sensors (Basel)* 2021; 21: 3472.
17. Gu Y, Wang YT, Wang M, et al. Secure user authentication leveraging keystroke dynamics via wi-fi sensing. *IEEE Trans Ind Inf* 2022; 18: 2784–2795.
18. Zhang Y, Zheng Y, Qian K, et al. Widar3.0: zero-effort cross-domain gesture recognition with wi-fi. *IEEE Trans Pattern Anal Mach Intell* 2022; 44: 8671–8688.
19. Feng C, Wang N, Jiang YC, et al. Wi-Learner: towards one-shot learning for cross-domain wi-fi based gesture recognition. *Proc Acm Interact Mob Wearable Ubiquitous Technol-IMWUT* 2022; 6: 27.
20. Zhang F, Chen C, Wang BB, et al. Wispeed: a statistical electromagnetic approach for device-free indoor speed estimation. *IEEE Internet Things J* 2018; 5: 2163–2177.
21. Guo L, Guo S, Wang L, et al. Wiar: a public dataset for wifi-based activity recognition. *IEEE Access* 2019; 7: 154935–154945.
22. Li W, Bocus MJ, Tang C, et al. On CSI and passive wi-fi radar for opportunistic physical activity recognition. *IEEE Trans Wireless Commun* 2022; 21: 607–620.
23. Wang J, Zhang L, Gao Q, et al. Device-free wireless sensing in complex scenarios using spatial structural information. *IEEE Trans Wireless Commun* 2018; 17: 2432–2442.
24. Yang J, Chen X, Zou H, et al. Sensefi: a library and benchmark on deep-learning-empowered WiFi human sensing. *Patterns (N Y)* 2023; 4: 100703.
25. Wang YX, Wu KS and Ni LM. Wifall: device-free fall detection by wireless networks. *IEEE Trans Mob Comput* 2017; 16: 581–594.
26. Wang H, Zhang D, Wang Y, et al. RT-Fall: a real-time and contactless fall detection system with commodity WiFi devices. *IEEE Trans Mob Comput* 2017; 16: 511–526.
27. Palipana S, Rojas D, Agrawal P, et al. Falldefi: ubiquitous fall detection using commodity wi-fi devices. *Proc ACM Interact, Mob, Wearable Ubiquitous Technol* 2018; 1: 1–25.
28. Yang X, Xiong F, Shao Y, et al. Wmfall: wiFi-based multi-stage fall detection with channel state information. *Int J Distrib Sens Netw* 2018; 14: 1550147718805718.
29. Ramezani R, Xiao Y and Naeim A. Sensing-Fi: wi-fi CSI and accelerometer fusion system for fall detection. In: 2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), 2018, pp.402–405.
30. Damodaran N, Haruni E, Kokhkharova M, et al. Device free human activity and fall recognition using WiFi channel state information (CSI). *CCF Trans Pervas Comput Interact* 2020; 2: 1–17.
31. Wang Y, Yang S, Li F, et al. Fallviewer: a fine-grained indoor fall detection system with ubiquitous wi-fi devices. *IEEE Internet Things J* 2021; 8: 12455–12466.
32. Shalaby E, ElShennawy N and Sarhan A. Utilizing deep learning models in CSI-based human activity recognition. *Neural Comput Appl* 2022; 34: 5993–6010.
33. Nakamura T, Bouazizi M, Yamamoto K, et al. Wi-Fi-based fall detection using spectrogram image of channel state information. *IEEE Internet Things J* 2022; 9: 17220–17234.
34. Xia Z and Chong S. WiFi-based indoor passive fall detection for medical internet of things. *Comput Electr Eng* 2023; 109: 108763.
35. Hu Y, Zhang F, Wu C, et al. Defall: environment-independent passive fall detection using WiFi. *IEEE Internet Things J* 2022; 9: 8515–8530.
36. Dayal S, Narui H and Deligiannis P. *Human fall detection in indoor environments using channel state information of Wi-Fi signals*. Redwood City, California: Stanford University, 2015.
37. Halperin D, Hu W, Sheth A, et al. Tool release: gathering 802.11n traces with channel state information. *ACM SIGCOMM Comput Commun Rev* 2011; 41: 53–53.
38. Halperin D, Ward D and Ziegler J. linux-80211n-csitool-supplementary, https://github.com/dhalperi/linux-80211n-csitool-supplementary (2015, accessed 18 December 2023).
39. Fard Moshiri P, Shahbazian R, Nabati M, et al. A CSI-based human activity recognition using deep learning. *Sensors (Basel)* 2021; 21: 7225.
40. Hanley JA and McNeil BJ. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 1982; 143: 29–36.
41. ONNX, https://github.com/onnx/onnx (2017, accessed 18 December 2023).
42. Forbes G. CSIKit: Python CSI processing and visualisation tools for commercial off-the-shelf hardware, https://github.com/Gi-z/CSIKit (2021, accessed 18 December 2023).
43. Hongshixian. CSI reader, https://github.com/hongshixian/CSI_reader (2020, accessed 18 December 2023).
44. Marowka A. Python accelerators for high-performance computing. *J Supercomput* 2017; 74: 1449–1460.
45. Numba, http://numba.pydata.org/ (2018, accessed 18 December 2023).
46. Lam SK, Pitrou A and Seibert S. Numba: a llvm-based python jit compiler. In: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC. 2015, pp.1–6.