mathematics

MDPI

*Article*

# User Authentication by Gait Data from Smartphone Sensors Using Hybrid Deep Learning Network

**Qian Cao [1,2,*], Fei Xu [1,2] and Huiyong Li [3]**

[1] School of E-Business and Logistic, Beijing Technology and Business University, Beijing 100048, China; 2130112050@st.btbu.edu.cn
[2] National Engineering Laboratory for Agri-Product Quality Traceability, Beijing Technology and Business University, Beijing 100048, China
[3] School of Computer Science and Engineering, Beihang University, Beijing 100191, China; lihuiyong@buaa.edu.cn
* Correspondence: caoqian@th.btbu.edu.cn

**Abstract:** User authentication and verification by gait data based on smartphones' inertial sensors has gradually attracted increasing attention due to their compact size, portability and affordability. However, the existing approaches often require users to walk on a specific road at a normal walking speed to improve recognition accuracy. In order to recognize gaits under unconstrained conditions on where and how users walk, we proposed a Hybrid Deep Learning Network (HDLN), which combined the advantages of a long short-term memory (LSTM) network and a convolutional neural network (CNN) to reliably extract discriminative features from complex smartphone inertial data. The convergence layer of HDLN was optimized through a spatial pyramid pooling and attention mechanism. The former ensured that the gait features were extracted from more dimensions, and the latter ensured that only important gait information was processed while ignoring unimportant data. Furthermore, we developed an APP that can achieve real-time gait recognition. The experimental results showed that HDLN achieved better performance improvements than CNN, LSTM, DeepConvLSTM and CNN+LSTM by 1.9%, 2.8%, 2.0% and 1.3%, respectively. Furthermore, the experimental results indicated our model's high scalability and strong suitability in real application scenes.

**Keywords:** gait recognition; inertial sensor; deep learning; smartphone

**MSC:** 68T07

**Citation:** Cao, Q.; Xu, F.; Li, H. User Authentication by Gait Data from Smartphone Sensors Using Hybrid Deep Learning Network. *Mathematics* **2022**, *10*, 2283. https://doi.org/10.3390/math10132283

Academic Editor: Fuyuan Xiao

Received: 28 May 2022
Accepted: 27 June 2022
Published: 29 June 2022



## 1. Introduction

Human gait, which is a manner of human walking or moving, consists of periodic signals with repetitive identical patterns [1,2]. Everyone has a unique gait, which makes it a suitable biometric trait for identify recognition. Unlike other biometric features, gait can be obtained from a long-distance without being noticed. Therefore, gait recognition has gradually drawn significant attention in the last decade [3–6].

According to different walking terrains and walking patterns, gait recognition methods can be divided into three main categories, computer vision-based gait recognition [7,8], floor sensor-based gait recognition [9] and inertial sensors-based gait recognition [10]. Computer vision-based gait recognition, in which subjects are recorded and analyzed through the use of video cameras, is well-advanced because of its large variety of different applications. However, there are still some limitations. Firstly, light and visual distance affects the accuracy of biometric gait recognition. Secondly, it is time- and money-consuming to deploy data acquisition equipment, which is indispensable in computer vision-based gait recognition. More importantly, the angle of vision formed

between the plane of the camera and the sagittal plane of the user is an extremely limiting factor, limiting its widespread application [11]. Floor sensor-based gait recognition, in which sensors are usually integrated directly into the floor or installed in mats, performs well in access control and smart homes. Yet, its performance is not very satisfactory in a continuous authentic system, since it has high requirements for locations. With the advent of the micro-electro-mechanical system (MEMS) technology, the application of inertial sensors is becoming increasingly mature [12]. Moreover, smartphones, which integrate many advanced inertial sensors, including accelerometers and gyroscopes, are widely used today due to their small size, portability and low power consumption [13–15]. Therefore, gait recognition based on smartphones' inertial sensors, in which data are generated by the movement of a walking body, has gradually attracted increasing attention for human identification and verification [16–21]. One application is the recognition of a smartphone's owner via the analysis of gait information collected by the smartphone's sensors. Additionally, the extraction of discriminative features from complex data collected from smartphones becomes an important issue. The latest advances in deep learning have prompted researchers to apply it to gait identification. In order to achieve accurate identity certification, existing methods often set a requirement that the subjects have to walk along a specified road and/or at a normal speed, which limits severely the scope of its application. In order to conduct identity recognition with loose constraints on where and how the users walk, we presented a more robust hybrid deep learning model for gait recognition (The source code can be seen at https://github.com/xfyy/GaitRecognitionForAndroid.git (accessed on April 15, 2022)).

The main contributions of this paper are as follows:

1. Firstly, we proposed a hybrid deep learning model (HDLN), which combined a LSTM network and a CNN to recognize gait for identity authentication. The signals are fed into the hybrid network to generate feature vectors. To improve the important features' proportion in multiple dimensions during decision-making, we optimized the convergence layer of the HDLN based on attention mechanism and spatial pyramid pooling. This algorithm optimizes the structure of the HDLN and thus improves the feature learning performance.
2. Secondly, we developed an APP to collect data under the unrestricted conditions on where and how the users walk, which gets rid of the restrictions on road conditions and walking speed.
3. Lastly, we proposed a novel data segmentation algorithm aiming at the problem of gait cycles produced in data segmentation. Data are segmented using the sliding window algorithm, and the similarity in the segmented gait data is estimated using the improved correlation algorithm to discard abnormal gait cycles.

The rest of the paper is organized as follows. Section 2 introduces the related work. Data preprocessing is described in Section 3. Section 4 introduces the proposed method in detail. Section 5 presents the experimental results, and discusses several optimizations for the user identification system. The conclusion is drawn in Section 6.

## 2. Related Work

Gait analysis-based biometric identification has attracted the attention of many researchers, prompting a great deal of research on the subject. In this section, we review the state-of-the-art work related to our study, which can be mainly categorized into two aspects, as follows.

### *2.1. Pattern Recognition for Gait Recognition*

In early research of inertial-based gait recognition, Ailisto et al. [22] first proposed gait recognition using inertial sensors. A trivial accelerometer attached to the user's waist was utilized to collect signals. They used a template matching-based solution by computing cross-correlation between extracted cycles and the stored template, obtaining an EER

of 6.4%. The research in [23] achieved the first scheme on smartphones-based gait recognition. They used Google G1 worn on the hip to acquire acceleration data and obtained EERs of 5% and 9% for histogram similarity and cycle length methods respectively. These two studies play an important role in the early development of gait recognition.

Aside from the above research, many other gait-recognition methods have been presented. Rong et al. [24] opted for a statistical-modeling solution rather than the previous template-matching options. They aimed to provide a solution that avoids knowledge of sensor location, which runs in real time on Android phones. In their paper, a sliding window of 512 samples with 50% overlap is used to segment gait cycles. Then a GMM–UBM (Gaussian Mixture Model—Universal Background Model) is trained to verify the users' identity. Finally, they obtained an EER of 14%, which provided further promotion for the development of gait recognition. In addition, the summary of the above studies is shown in Table 1.

**Table 1.** The summary of the pattern recognition studies for gait recognition.

| Method | Methodology | Performance | Limitation |
| --- | --- | --- | --- |
| Ailisto et al. [22] | template matching and cross-correlation computation | 6.4% EER | data collected under limited conditions |
| Derawi et al. [23] | histogram similarity and cycle length methods | 5% and 9% EER | data collected under limited conditions |
| Rong et al. [24] | GMM-UBM | 14% EER | data collected under limited conditions |

### *2.2. Deep Learning for Gait Recognition*

With deep learning growing at a fast pace, CNNs and Recurrent Neural Networks (RNNs) are gradually replacing the previous methods for feature learning in gait recognition. CNNs are used to process array signals, such as images, and have been verified as a successful feature extractor in image recognition. RNNs are employed to process input signals for time series. Accelerometer and gyroscope signals could generally be processed into two-dimensional time series [25]. Therefore, CNNs can represent inertial data through convolutional feature maps, while RNNs can utilize their advantages by processing the gait data as a time series.

The first attempt to use deep learning for feature extraction was the approach proposed by Gadaleta and Rossi [18]. They obtained motion signals, including accelerometer and gyroscope, from smartphones. In their research, by treating the gait matrix as an image, the employment of CNN as a feature extractor was first proposed to avoid the subjectivity of manually selecting statistical features. Finally, they obtained a misclassification rate less than 0.15% when a One-class Support Vector Machine (OSVM) is used as an authentication method. Additionally, their comprehensive experiments proved CNNs' great features learning performance on gait recognition. However, CNNs tend to ignore the temporal information of human activities. Therefore, many variants of CNNs are used to solve these problems. In addition, RNNs as favorable tools for processing time series data are gradually introduced into inertial sensor-based gait recognition.

In [26], a RNN was used to generate feature vectors in the Osaka University Database (OUDB) for gait recognition. The researchers explored the effect of different LSTM's parameters on gait recognition. Then, they chose the proper parameters for classification, and compared with other algorithms, which were previously tested on the same database, and finally found RNN can effectively extract gait features. Bari et al. [27] proposed a CNN KinectGaitNet for Kinect-based gait recognition, in which the 3D coordinates of each of the body joints over the gait cycle are transformed to create a unique input representation. It outperforms all state-of-the-art methods for Kinect-based gait recognition. Huang et al. [28] proposed a lightweight attention-based CNN model, which en-

hanced the distinctness of the extracted gait features and reduced the network parameters. Additionally, they would further implement the gait recognition model on a smartphone.

In recent papers, many researchers began to adopt hybrid methods for gait recognition studies. Zou et al. [19] proposed a combined LSTM and CNN method for feature learning, and reported higher accuracy in person identification and authentication. The research in [29] proposed a gait segmentation algorithm, which combined the loading information captured by strain gauges with angular velocity data to reliably and accurately segment gait events. The experimental results verified the ability of this algorithm. Unlike this research, our approach is not so strict in respect of segmentation granularity since we focus on identification recognition according to different gait data from different persons.

Existing research has demonstrated the effectiveness of deep learning in gait recognition; however, there are still several insufficiencies. First, during data acquisition, many gait collection factors have been considered. Yet most researchers focused on data collected under limited road conditions with specified walking speed, or with the smartphones placed in a fixed position. The identification of gait in unconstrained environments, freely timed or with unconstrained speed, remained challenging. Second, in feature representation, the previous research extracted features using the same weights in the decision-making phase. Understanding how to extract more efficient gait features for gait recognition was still unknown. Some of the studies that applied deep learning for gait recognition are listed in Table 2.

**Table 2.** The summary of the deep learning studies for gait recognition.

| Method | Methodology | Performance | Limitation |
| --- | --- | --- | --- |
| Gadaleta and Rossi [18] | CNN | 0.15% misclassification rate | Only using CNN and data collected under limited conditions |
| Fernandez-Lopez et al. [26] | LSTM | 7.55% EER | Only using LSTM and data collected under limited conditions |
| Zou et al. [19] | CNN + LSTM | gait recognition accuracy with 93.75% | Using original CNN + LSTM with no improvements |

Aiming to overcome the shortcomings of previous gait recognition research, we proposed a method with unrestricted conditions of where, when and walking speed for data collection. A hybrid network with an improved convergence layer used simultaneously for feature representation and classification is employed, making for a more practical and more accurate approach.

## 3. Data Preprocessing

Data preprocessing includes data resampling, denoising and gait cycle extraction. These preprocessing methods reduce the errors caused in data acquisition, as well as the amount of calculation required.

### *3.1. Gait Data Collection and Preprocessing*

We developed an Android application using Android Studio, and then installed it to smartphones of different brands, Huawei Mate 9, Vivo Y17 and Samsung S6.

Data acquisition includes two parts, and the first part is the development of an application to collect data. During application development, we used three Android components, Activity, Service and Broadcast Receiver [30]. Activity represents the user interface, which is used to display the data from accelerometers and communicate with

Service component. Service ensures that data collection is continuously performed in the background. Broadcast Receiver, which is the communication bridge between system and users, is responsible for the timely transmission of the collected user gait data to the user interface. The background interaction of the acquisition software is shown in Figure 1.

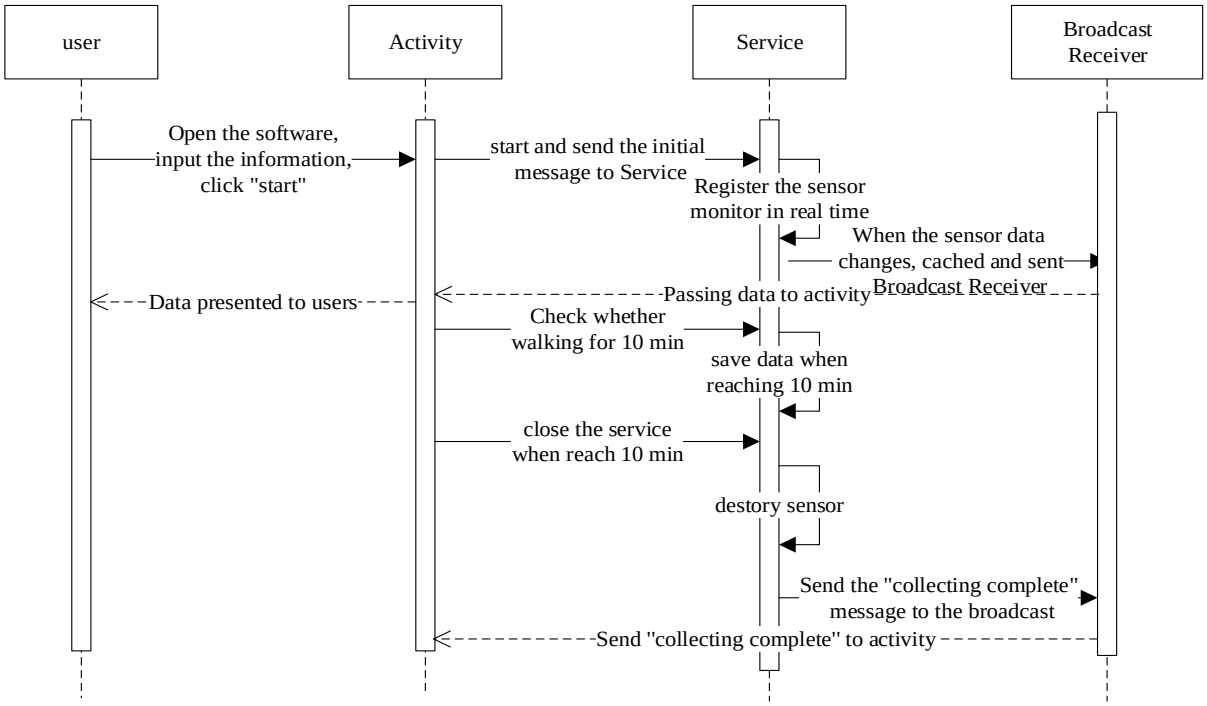


**Figure 1.** The background interaction process of the acquisition software.

The second part is to collect gait data in the conditions of unlimited roads, time or speeds, using the application we developed. Data were collected in daily life, such as during a walk after a meal. A hall with an area of 10 × 25 square meters, a playground, and so on, were all optional experimental sites. The sampling frequency was 50 Hz, and the sampling time of each volunteer was 10 min, with each volunteer sampled 10 times. Accelerometer and gyroscope gait data were collected from 40 subjects, including 27 males and 13 females, aged 14 to 56 years old. None of the participants had any gait abnormalities. Android smartphones were put in the right front pocket of the participants' trousers without direction restriction. The participants walked in their own manner without any special training or in a certain order, but all walked on the same path, since gait changed greatly when participants walked on different paths. We exported data in .txt format. The .txt folder contained a total of 8 columns, including time, acceleration in the x, y and z axes directions, gyroscope in the x, y and z axes and user name. The software interface for data collection is shown in Figure 2. In addition, the accelerometer recorded the movements of the device along the three axes of the mobile phones' coordinate system, the orientation sensor was used to detect the mobile phones' direction and the gyroscope measured the rotation along the three axes of the mobile phones' coordinate system. Before walking, the preparation time was set to 20 s, and the walking time was set to 10 minutes to obtain enough data.

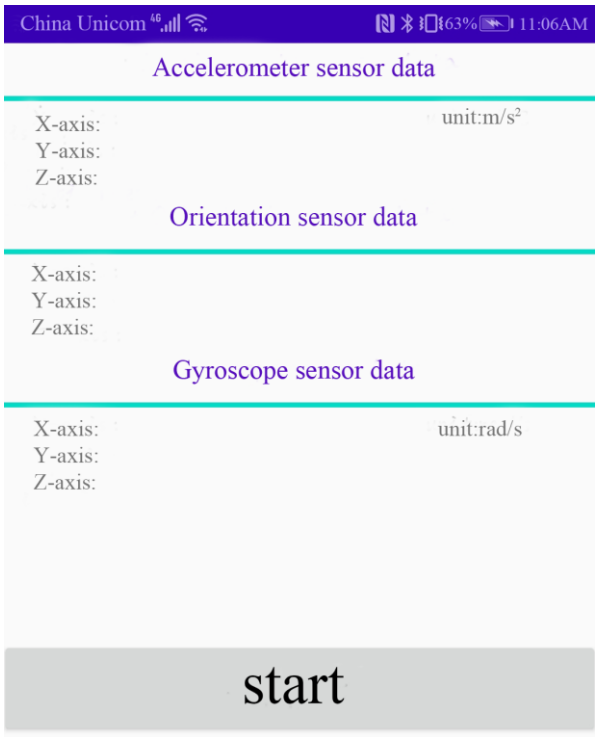


**Figure 2.** Software interface for data collection.

### *3.2. Noise and Impact of Orientation Elimination*

Due to the limitations of the Android SDK [31], data cannot be recorded at a fixed sampling rate, which results in different time intervals between consecutive records. To solve this problem, we first resampled data with a frequency of 100 Hz. Then a low pass finite pulse response FIR (final impulse response) filter with 8 Hz was used to denoise the resampled data to reduce the motion artifacts which may have appeared at higher frequencies.

The problem of device orientation instability was addressed by an Orientation Independent Transformation algorithm [32]. The basic principle of this algorithm is to obtain a new reference system with three orthogonal coordinate axes $f_1$, $f_2$ and $f_3$. $f_1$ represents the upward direction (parallel with a subject's body). $f_2$ points forward (consistent with the direction of movement) and $f_3$ is orthogonal to $f_1$ and $f_2$.

Assuming that $n_k$ is the number of samples in the current walking cycle $k$ ($k$ = 1, 2, 3, ……), $A=[a_x, a_y, a_z]^T$ with size of $3\times n_k$ describes the acceleration matrix while $G=[g_x, g_y, g_z]$ with the same size as $A$ represents the gyroscope matrix. $a_x$, $a_y$, and $a_z$ are used to represent acceleration samples along the $x$, $y$, and $z$ axes where $|a_x|=|a_y|=|a_z|=n_k$. Meanwhile, $g_x$, $g_y$, and $g_z$ indicate gyroscope samples in the same cycle $k$, where $|g_x|=|g_y|=|g_z|=n_k$.

The specific steps are shown as follows. Firstly, we chose the center of gravity as the starting point [33]. Therefore, the mean gravity direction in the current walking cycle was the new reference system's first axis. The mean direction of gravity $\rho$ in period $k$ is evaluated as Equation (1).

$$\rho=(\overline{a_x}, \overline{a_y}, \overline{a_z}) \tag{1}$$

The first coordinate axis $f_1$ can be calculated by Equation (2).

$$f_1 = \frac{\rho}{\|\rho\|} \tag{2}$$

The projection of matrices $A$ and $G$ on the axis $f_1$ are obtained by Equation (3).

$$a_{f_1} = A \cdot f_1, g_{f_1} = G \cdot f_1 \tag{3}$$

Secondly, as shown in Equation (4), we obtain $A^f = [a_x^f, a_y^f, a_z^f]^T$, which is horizontal acceleration. Then we determine the direction of $f_2$, and the coordinate axis can be obtained by Equation (5), where $v$ is the direction of the maximum variance of $A^f$ and calculated by Principal Component Analysis (PCA).

$$A^f = A - f_2 a_{f_2}^T \tag{4}$$

$$f_2 = \frac{v}{\|v\|} \tag{5}$$

Then data from accelerometer and gyroscope are projected on $f_2$ according to $a_{f_2} = A \cdot f_2$ and $g_{f_2} = G \cdot f_2$. Lastly, $f_3$ is obtained by an across product shown in Equation (6) as these three coordinate axes are orthogonal.

$$f_3 = f_1 \times f_2 \tag{6}$$

Along $f_3$ axis, the new accelerometer and gyroscope data are $a_{f_3} = A \cdot f_3$ and $g_{f_3} = G \cdot f_3$.

By data preprocessing, the influence of the device's orientation on the gait signals is eliminated, and all gait signals are represented in a fixed coordinate system. Figure 3 illustrates the acceleration before and after orientation elimination. Obviously, the characteristics of sensor signals in different positions are more similar by using orientation elimination.

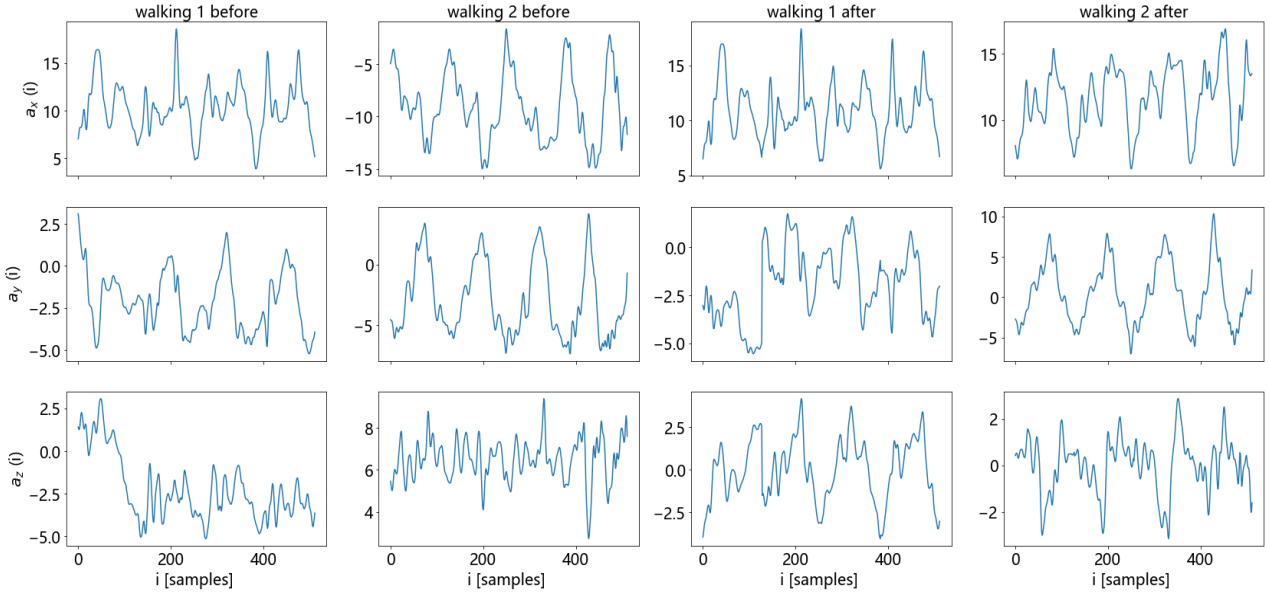


**Figure 3.** Comparison of accelerations before and after orientation elimination. All the data are collected from one person. For the three subgraphs in each column, the horizontal axis represents time, and the vertical axis represents the projection of acceleration in x, y and z directions. The data in the first and the second column are collected in two different positions respectively, while the data in the last two columns are collected after orientation elimination.

*3.3. Gait Cycle Segmentation*

In the following step, we divide the continuous data into separate segments, which refers to touching the ground with the same foot twice in succession. In this paper, we use sliding window [34] with a fixed length without overlapping to segment gait cycles since this does not destroy the morphology of gait patterns. Additionally, the abnormal data segments are discarded by calculating the similarity between data segments.

The window segmentation method can be denoted as follows: assuming the sample matrix is *S*, the window size is *m*, the step size of the window is *step*, and the window segmentation algorithm is shown in Equations (7) and (8).

$$s_1 = \begin{bmatrix} a_1(t_i) & a_2(t_i) & \dots & a_m(t_i) \\ a_1(t_{i+1}) & a_2(t_{i+1}) & \dots & a_m(t_{i+1}) \\ \dots & \dots & \dots & \dots \\ a_1(t_{i+m}) & a_2(t_{i+m}) & \dots & a_m(t_{i+m}) \end{bmatrix} \tag{7}$$

$$s_2 = \begin{bmatrix} a_1(t_{i+step}) & a_2(t_{i+step}) & \dots & a_m(t_{i+step}) \\ a_1(t_{i+step+1}) & a_2(t_{i+step+1}) & \dots & a_m(t_{i+step+1}) \\ \dots & \dots & \dots & \dots \\ a_1(t_{i+step+m}) & a_2(t_{i+step+m}) & \dots & a_m(t_{i+step+m}) \end{bmatrix} \tag{8}$$

where $s_1$ and $s_2$ represent two adjacent data windows with size *m* extracted from the original sequence using a window segmentation method. Generally speaking, the larger the value of *m*, the more the sequence period is involved, and the better the classification is in theory.

Similarity calculation, as shown in Equation (9), is used to discard the abnormal gait cycles. During similarity calculation, we take the first segment of the data set of the same type as the standard for similarity calculation.

$$V(i) = \frac{(Z - \overline{Z}1_N)^T \times (C - \overline{C}1_N)^T}{\left\| Z - \overline{Z_N} \right\| \left\| C - \overline{C_N} \right\|} \tag{9}$$

where vectors $Z$ and $C$ have the same size of *N*. $C$ is the comparison template and $Z$ is the gait segment needs to be compared. $\overline{Z}$ and $\overline{C}$ represent respectively the average value of $Z$ and $C$. Vector $1_N = (1,1,...,1)^T$ with $|1_N| = N$. $\|\cdot\|$ is the L2-norm operator. $V(i)$ not less than 0.5 is saved as gait cycles.

After the gait cycles are divided, they can be represented as $X = (a_{f1}, a_{f2}, a_{f3}, g_{f1}, g_{f2}, g_{f3})$. Then, the divided gait cycles can be used as the inputs of HDLN.

## 4. Methodology

The framework of the proposed HDLN gait recognition method is shown in Figure 4. It consists of two LSTM layers, L1 and L2, and the number of LSTM units is 64. There are three one-dimension convolutional layers: C1, C2 and C3, two pooling layers: P1 and P2, a spatial pyramid pooling layer: SSP, an attention layer: A1 and a softmax layer. The output of layer L2 and SSP are combined as the features extracted by CNN and LSTM. Before entering the softmax layer, the attention layer is used for feature combination.

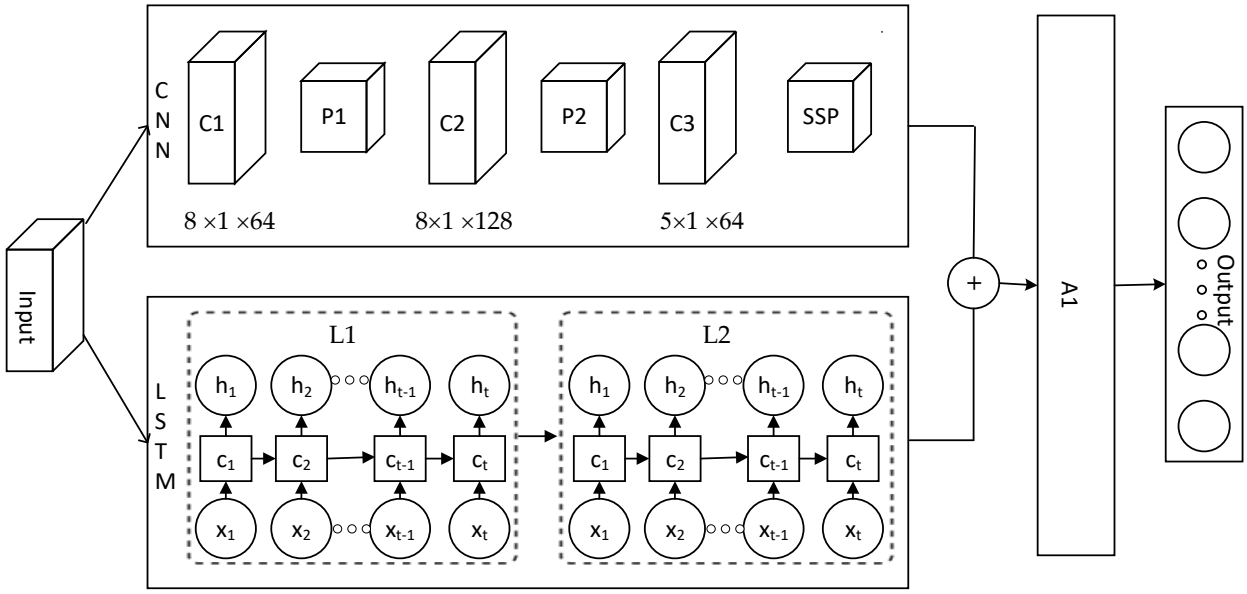


**Figure 4.** The framework of HDLN.

The first part of HDLN is a LSTM network, a special RNN, which is designed to solve the long-term dependency problem. Compared with fundamental RNN, LSTM has more robust memory capacity and performs better in more extend sequence signal data.

For a LSTM network with *L* hidden layers, a state $h_t^l$ will be generated as follows for each layer at time *t* with a gait sequence $x = (x_1, x_2, ..., x_T)$:

$$h_t^l = \sigma(w_{xh}^l x_t + h_{t-1}^l w_{hh}^{tl} + h_t^{l-1} w_{hh}^{ll} + b_h^l) \tag{10}$$

where $h_t^l$ represents the state of layer *l* at time t, and $x_t$ is input at time *t*. The weight matrix of the input $x_t$ to the *l*-th hidden layer is represented by $w_{xh}^l$, and $w_{hh}^{tl}$ is the weight matrix of the state at time $t-1$ to the state at time *t* at the same layer *l*. $w_{hh}^{ll}$ is the weight matrix of the state at layer $l-1$ to the state at layer *l* at the same time t, and $b_h^l$ is the bias of layer *l*, and σ (·) is the activation function.

The core of LSTM is the cell, and three gates, forget gate, input gate and output gate, control the state of cell. Similar to RNN, the state of LSTM network can be represented by $h_t^l$. The input gait *i* refers to the information adding to cell state, and forget gate *f* is to decide what information cell state needs to be discarded. Output gate *o* controls what information can be output. Input gate *i*, forgetting gate *f*, state vector *c*, output gate *o* and $h_t^l$ can be updated respectively according to Equations (11)–(15):

$$i_t = \sigma_i(w_{xi} x_t + w_{hi}^t h_{t-1}^l + w_{hi}^l h_t^{l-1} + w_{ci} c_{t-1} + b_i) \tag{11}$$

$$f_t = \sigma_f(w_{xf} x_t + w_{hf}^t h_{t-1}^l + w_{hf}^l h_t^{l-1} + w_{cf} c_{t-1} + b_f) \tag{12}$$

$$c_t = f_t + c_{t-1} \sigma_i(w_{xc} x_t + w_{hc}^t h_{t-1}^l + w_{hc}^l h_t^{l-1} + b_c) \tag{13}$$

$$o_t = \sigma_o(w_{xo} x_t + w_{ho}^t h_{t-1}^l + w_{ho}^l h_t^{l-1} + w_{co} c_{t-1} + b_o) \tag{14}$$

$$h_t^l = o_t \sigma_h(c_t) \tag{15}$$

where $w$, $\sigma$, $i_t$, $f_t$, $c_t$, and $o_t$ are the parameters of layer $l$. $w_{xi}$ is the weight matrix of the input $x_t$ to the input gate. $\sigma_i$ is the input gate's activation function, and $b_i$ is the bias of the input gate. $w_{xi}$, and $W$, $b$, and $\sigma$ can be inferred accordingly. Provided the LSTM network is constructed with $L$ hidden layers, with each containing $N$ hidden nodes, for each input $x = (x_1, x_2, ..., x_T)$, the output feature can be obtained by Equation (16).

$$fout_{lstm} = h_T^L = (f_1, f_2, ..., f_N) \tag{16}$$

The second part of the LSTM is a CNN, which consists of convolutional layers, non-linear layers and pooling layers. The convolutional layer applies a sliding kernel to the inputs and extracts features. The pooling layer reduces the feature points and better retains the basic properties of the feature points. The nonlinear layers apply an activation function on the features in order to enable the modeling of non-linear functions by the network. Compared with fully connected feed-forward neural networks, CNN has the advantages of weight shared and less computational complexity [35]. Meanwhile, CNN has been proved to be a successful feature extractor in the image field.

### *4.1. Convergence Layer*

In actuality, not all the extracted gait features are useful for gait recognition, therefore we applied an attention mechanism to extract the important gait information and ignore the unimportant information. Moreover, in order to extract features from more dimensions, we added spatial pyramid pooling [36] before the attention layer. Finally, the vectors of these information were combined as the output to focus on the key feature points. Through the spatial pyramid pooling and attention layer, more complete feature points, which will be used in identification operations, are extracted.

Assuming that $f = (f_1, f_2, ..., f_D)$ denotes the feature map extracted by CNN, and $f_1$ is one of the feature vectors, we first put the feature vectors into spatial pyramid pooling and the corresponding output of spatial pyramid pooling is calculated by Equation (18):

$$fout_{cnn} = \bigcup_{k=1,s=1}^{n} (C_{k\times 1,s}) = [C_{1\times 1,s}, C_{2\times 1,s}, C_{4\times 1,s}] \tag{17}$$

where $k$ is the kernel of the spatial pyramid pooling layer and $s$ is the number of steps. U (*) is the aggregation of the features sampling. $fout_{cnn}$ is the output value of spatial pyramid pooling. After $fout_{cnn}$ is obtained, we combine $fout_{cnn}$ and $fout_{lstm}$ as the final feature vectors $fout$. Then we use attention mechanism to extract the important gait information. When using attention mechanism, the hidden variables of each feature needs to be calculated as Equation (18).

$$\beta_i = V^T \tanh(w_i\, fout(i) + b_i) \tag{18}$$

where $\beta_i$ is the hidden variable, and $w_i$ and $b_i$ are the weight and bias of attention layer. $V^T$ is the parameter matrix. After the hidden variables are obtained, Equation (19) is used to transform the hidden variables exponentially to calculate the attention value $\alpha_i$ of each feature $fout_i$. The larger the value $\alpha_i$ is, the more attention is allocated, and the more important role the feature vector plays in gait classification.

$$\alpha_i = \frac{\exp(\beta_i)}{\sum_{i=1}^{n} \exp(\beta_i)} \tag{19}$$

Finally, the output vector of attention model is calculated as Equation (20):

$$\varepsilon = \sum_{i=1}^{n} \alpha_i f_i \tag{20}$$

*4.2. Softmax Layer*

The calculated feature map is then passed to a fully connected layer followed by a softmax function, and the final output is the probability distribution over all categories. Softmax function is shown in Equation (21):

$$Q_i(o) = \frac{e_{o_i}}{\sum_{i=1}^{m} e_{o_i}} \tag{21}$$

where $o_i$ is one of the features of A1, and $Q_i(o)$ is the softmax value of a certain user's input. Furthermore, Cross entropy is used as the loss function to measure the deviation between the actual value and the predicted value, and the loss is reduced by iteration.

## 5. Experimental Results and Discussion

*5.1. Experiment Configuration*

The experimental environment was set as follows: Intel Core i5-8250 U CPU @ 1.6 GHZ, 8 GB memory, Windows 10 operating system. The example was calculated in PyCharm, and TensorFlow 2.0 was employed. The forty volunteers mentioned in Section 3.1 participated in the experiments. The detailed gait data collection is given in the former gait data collection and preprocessing part. We repeated the experiments ten times and reported the average results to avoid accidental contributing factors.

*5.2. Experiment Results*

In order to evaluate the recognition accuracy of different algorithms, we define accuracy in Equation (22).

$$accuracy = \frac{the\ number\ \text{of}\ correctly\ classified\ gait\ cycles}{the\ number\ \text{of}\ total\ testing\ gait\ cycles} \tag{22}$$

5.2.1. Accuracy Comparison of Different Models with Our Dataset

To show the user identification accuracy of different models, we compared the HDLN with several advanced networks using our dataset, and the results are listed in Table 3. 70% of the volunteers were for training and the remaining were for testing and validation. Meanwhile, several experiments were carried out by varying the samples for training and testing. It became obvious that the HDLN we proposed obtains the best performance. In particular, it has higher accuracy than CNN, LSTM, CNN+LSTM [20] and DeepConvLSTM [21] by 1.9%, 2.8%, 1.3% and 2.0%, respectively.

In addition, we evaluated the runtime overhead for all these algorithms, as listed in Table 3. It can be easily seen that, for each algorithm, the average recognition time for these forty participants is 3.07 s, 3.12 s, 3.09 s, 2.98 s and 2.92 s, respectively, which indicates that our method is computationally effective.

**Table 3.** User identification accuracy with different advanced networks.

| Algorithm | Accuracy | Runtime Overhead (s) |
|---|---|---|
| CNN | 94.03% | 3.07 |
| LSTM | 93.17% | 3.12 |

| | | |
|---|---|---|
| DeepConvLSTM | 93.91% | 3.09 |
| CNN+LSTM | 94.52% | 2.98 |
| Our model | 95.79% | 2.92 |

5.2.2. Robustness Verification for Our Model on Different Datasets

To verify the robustness of our model, we conducted experiments on datasets IDNet [18] and whuGAIT [19]. For whuGAIT, we used both dataset #1 and dataset #2, which were collected in real scenarios, to evaluate our model.

The identification accuracy of HDLN on different datasets is shown in Table 4. We can see that the accuracy in the IDNet dataset can even reach 99.65%. This is because IDNet is collected from standard walking style, which makes the classification less challenging. For whuGAIT, whether dataset #1 or dataset #2, its accuracy is higher than that of the original method [19]. We can also conclude that HDLN is not only suitable for our own dataset, but also for datasets IDNet and whuGAIT. Furthermore, the experimental results indicate that our model is more practical in real applications.

**Table 4.** The robustness of HDLN on different datasets.

| Dataset | Accuracy |
|---|---|
| IDNet | 99.65% |
| whuGAIT (dataset #1) | 94.59% |
| whuGAIT (dataset #2) | 97.89% |

*5.3. Discussion*

We conduct further experiments to explore the affecting factors on our method. Three main factors, multi-sensor fusion, orientation transformation and hybrid network optimization, are discussed, respectively, as follows.

5.3.1. The Fusion of Accelerometer and Gyroscope

Figure 5 shows the gyroscope signals extracted from three randomly selected volunteers. The upper, middle and bottom signals are obtained, respectively, from volunteers No. 1, No. 2 and No. 3. The blue, yellow and green lines in each graph represent the components of the gyroscope data along $x$, $y$ and $z$ axes, respectively. Obviously, the curves of these three signals are each significantly different from the other.

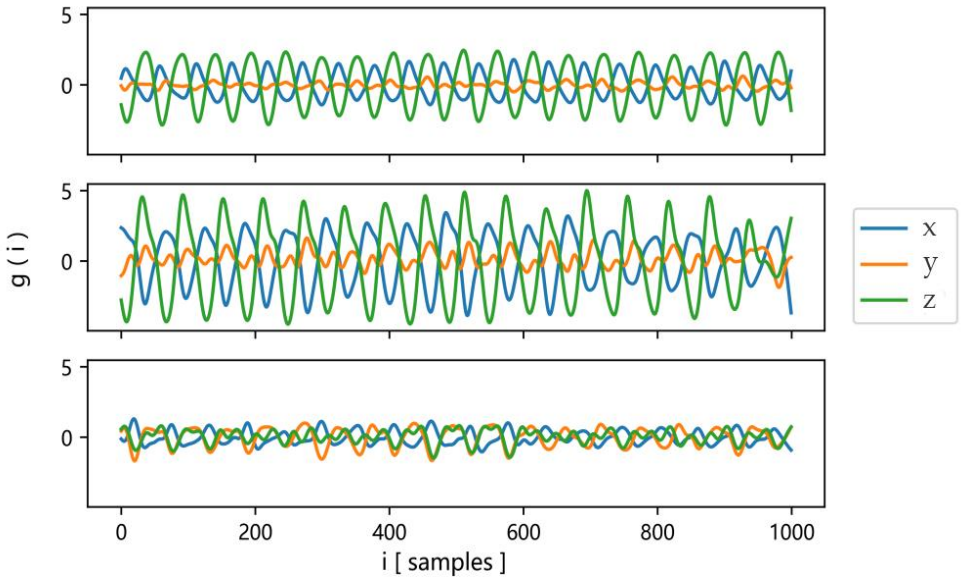


**Figure 5.** Comparison of gyroscope signals from three randomly selected volunteers.

First, in terms of amplitude, the max amplitude of volunteers No. 1, No. 2 and No. 3 range respectively from −2.0 to 2.0, −5.0 to 5.0 and −1.0 to 1.0. Second, these three volunteers' gyroscope curves are different, with the wave peaks and troughs in different positions. Meanwhile, the differences are also observed from other volunteers. It indicates that gyroscope data also can be used to differentiate between people.

In order to test the accelerometer and gyroscope data fusion on gait recognition, we compared the accuracy of using accelerometer or gyroscope alone against the combination of both, and the results are illustrated in Figure 6. The accuracy of using accelerometer data or gyroscope data alone are 94.78% and 93.10%, respectively, while the accuracy of their fusion is 95.79%. Apparently, the fusion of gyroscope and accelerometer data provides further improvements than using gyroscope or accelerometer alone.

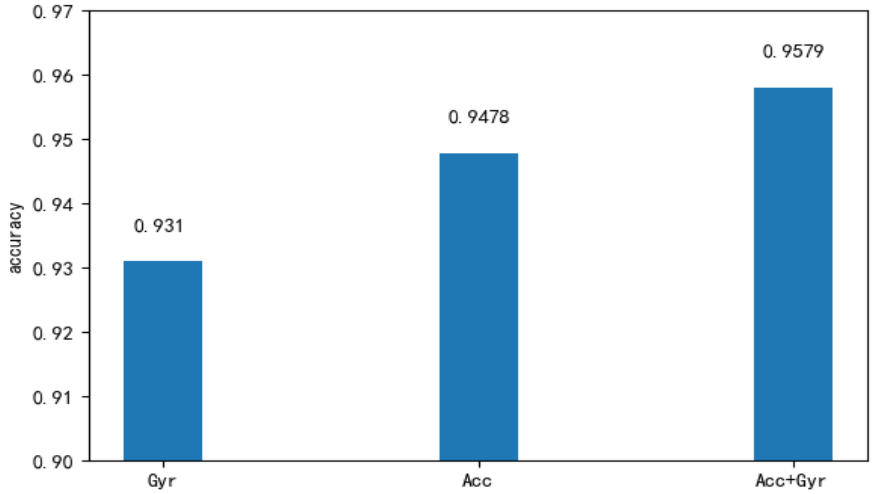


**Figure 6.** Accuracy comparison of using sensors fusion or not. Gyr represents using gyroscope alone, and Acc represents using accelerometer alone, and Acc + Gyr means the fusion of both.

5.3.2. The Effect of Orientation Transformation

To Table 5 shows the accuracy with or without the orientation transformation algorithm as a function of *Nc*. *Original* represents without the algorithm, and *Transformed* is with orientation transformation algorithm. It can be seen that the accuracy with orientation transformation is always higher than without orientation transformation. At the beginning, the accuracy increases dramatically as *Nc* increases. When *Nc* reaches 40, the accuracy gradually stabilizes. The accuracy using this algorithm can be improved by 0.1% when *Nc* reaches 80. To sum up, the orientation transformation algorithm improves the accuracy of gait recognition to some extent.

To compare the accuracy with or without the orientation transformation algorithm, we conducted experiments on the user identification system using our data.

**Table 5.** User identification accuracy with and without transformation.

| *Nc* | *Original* | *Transformed* |
|---|---|---|
| 10 | 90.45% | 91.86% |
| 20 | 92.76% | 93.88% |
| 30 | 93.54% | 94.23% |
| 40 | 93.81% | 94.28% |
| 50 | 94.41% | 94.62% |
| 60 | 94.41% | 94.67% |
| 70 | 94.51% | 94.67% |
| 80 | 94.62% | 94.72% |

5.3.3. Hybrid Network Optimizations

In this section, we propose some hybrid network optimizations, quantifying the classification performance.

First, we first discuss the influence of hyperparameters on a HDLN. Notably, a random grid hyperparameter selection is applied to find the hybrid parameters with the best performance.

In a CNN, the size of the convolution kernel directly affects the performance of the model. In this study, we explored a HDLN's accuracy against using different filter sizes in convolutional layer C1, C2 and C3. Figure 7 illustrates a HDLN's accuracy under different kernel sizes with C1, C2 and C3. It can be concluded that a HDLN achieves the highest accuracy under the condition of C1 with kernel size of 8, C2 with kernel size of 8 and C3 with kernel size of 5.

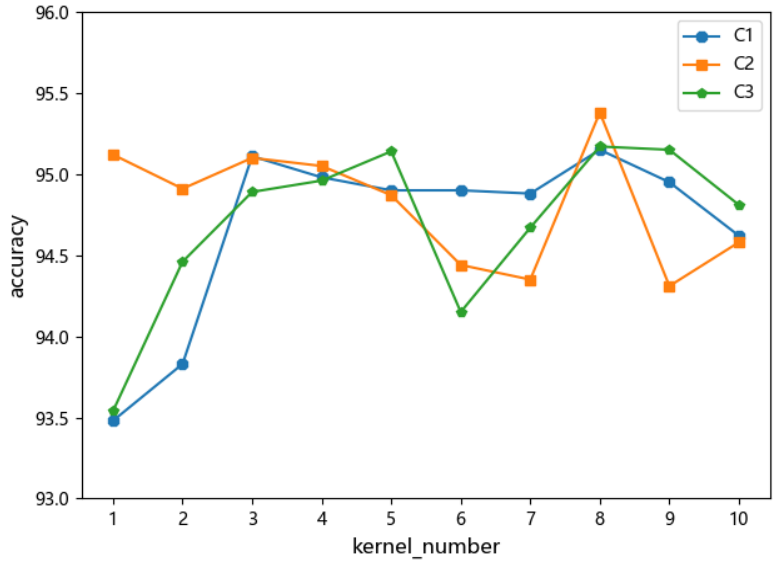


**Figure 7.** HDLN's Accuracy under different kernel number.

For a LSTM, the parameters we choose are listed in Table 6. We adopt a random grid hyperparameter selection followed by a hand-tuning method to determine the final hyperparameter, and the result is shown in Figure 8. It is apparent that the accuracy is the lowest when the number of LSTM units is 16, however, as the number of a LSTM cell increases, the accuracy does not always increase. The optimal number of LSTM layers is two, therefore we chose two LSTM layers with 64 units each.

**Table 6.** LSTM hyperparameter random grid.

| Variable | Values |
|---|---|
| Number of LSTM layers | [1, 3] |
| Number of Fully Connected Layers | [0, 3] |
| Number of filters | $[2^1, 2^8]$ |
| Feature Vector size | $[2^1, 2^8]$ |

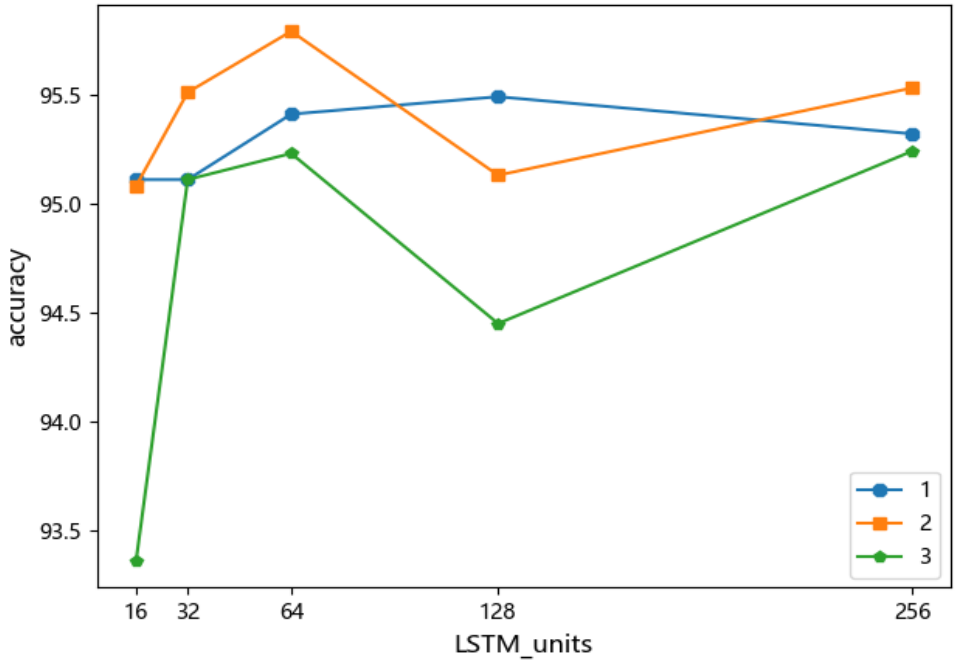


**Figure 8.** Accuracy with different LSTM units and layers. LSTM_units represent the number of LSTM units and curves of different colors represent different layers.

**How to combine CNN and LSTM:** Five hybrid networks which combine different models are compared to explore the best combination of CNN and LSTM. The different combinations are described as follows.

1. CNN: It has been trained from scratch.
2. LSTM: It is also trained from scratch.
3. (CNN+LSTM)_hor: CNN and LSTM are respectively trained from scratch. Then the output of CNN and LSTM are concatenated in horizontal.
4. (CNN+LSTM)_ver: CNN and LSTM are respectively trained from scratch. Then the output of CNN and LSTM are concatenated in vertical.
5. fix_CNN+LSTM: CNN is reconstructed by pre-trained weight parameters, while LSTM is trained from scratch, and their combination is vertical.
6. CNN+fix_LSTM: LSTM is reconstructed by pre-trained weight parameters. CNN is trained from scratch, and their combination is vertical.

The performance of different combinations is demonstrated in Figure 9. It is shown that CNN outperforms LSTM approximately by 1.1% in terms of accuracy. CNN+LSTM doesn't perform as well as CNN only, and one reason is that CNN+LSTM makes hybrid network too complex to be fully trained. In other words, during the training process, the hybrid network trained from scratch may suffer from over-fitting. Moreover, we can see that when a pre-trained network is combined with another one trained from scratch, the accuracy is higher than CNN or LSTM alone. Therefore, using the pre-trained network will be helpful for improving performance. The method, which sets the weight parameters of one network to fixed values and trains the other, reduces the training complexity and takes advantage of CNN and LSTM as well. Furthermore, it can be seen that the CNN+fix_LSTM network achieves the best performance. Therefore, we choose CNN+fix_LSTM as the final structure to build our model.

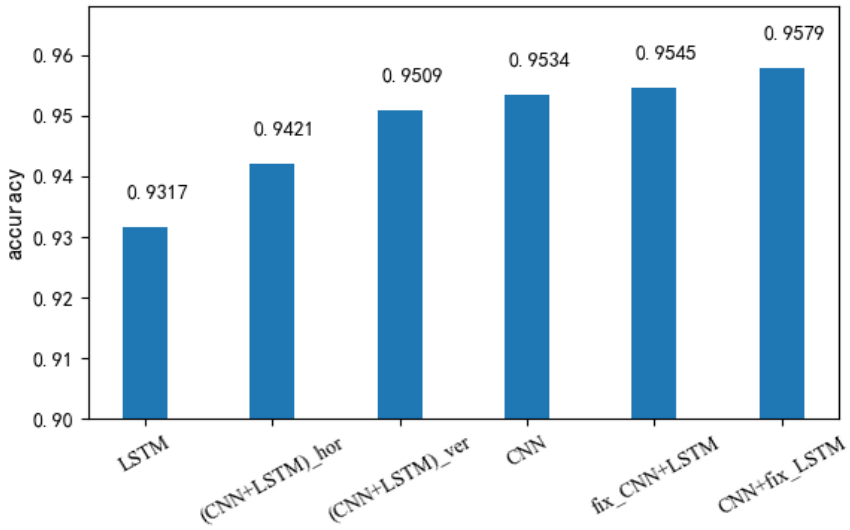


**Figure 9.** Accuracy of different network combinations.

The learning rate directly affects the ability of the model to find the optimal solution. If the learning rate is set to very low, the model parameters will update very slowly, requiring a long training time. On the contrary, if the learning rate is too large, it will lead to the oscillation of the loss function, an inability to converge, and poor classification capability. Table 7 shows that the accuracy is highest when the learning rate is 0.001 in this experiment.

**Table 7.** Accuracy of different learning rates.

| Learning Rate | Accuracy |
|---|---|
| 0.001 | 93.47% |
| 0.005 | 93.01% |
| 0.01 | 92.67% |
| 0.1 | 89.27% |

## 6. Conclusions and Future Work

This article presented a new approach to gait recognition system for smartphones by introducing the hybrid learning model (HDLN). We created an Android application to collect gait signals without scene and time limitations. Compared to the previous feature-learning methods, HDLN exploits the advantages of the CNN and LSTM, and improves the coverage layer. From the experimental results, we can draw the following conclusions: (1) the method with both accelerometers and gyroscopes can obtain better performance than accelerometers or gyroscopes alone. (2) The orientation transformation improves the accuracy of user gait recognition through a reduction in the influence of sensor directions. Additionally, an appropriate cutoff frequency in the gait recognition obtains a better performance. (3) HDLN obtains performance improvements when compared to other competing networks in user identification, providing a promising method for gait recognition.

Our work has more potential applications in practical applications: (1) Phone security. While a user walks, the smartphone would record the movements. If a stranger picks up this phone, the phone will verify the identity of the person who is holding it. If the verification fails, the phone will produce warnings. Therefore, it provides an additional layer of identity verification on top of fingerprint recognition and face recognition. (2) Medical field. It could also work with health care systems, adding an additional value to the access system, helping to distinguish user spoofing. (3) Gender recognition is a new

application that uses gait characteristics. It has become more and more popular in forensic and medical fields in recent years. For example, gender identification based on gait recognition can be used as a recommendation system for videos, products and applications. (4) Gait biometric attacks and countermeasures need further research. Under normal circumstances, it is difficult to deceive gait recognition system. Yet, in some special cases, for example, with the help of a specially designed treadmill, gait biometrics may be easily deceived. The false acceptance rate even reaches roughly 70%. Therefore, the effective avoidance of malicious attacks will be a challenging task for gait recognition.

Of course, our work still has limitations. We applied gait cycle segment points to divide gyroscope signals under the assumption that the timestamps of the accelerometer and gyroscope are synchronous. However, this does not work in some relevant use cases due to the Android system's time delay. We will address this problem in future studies on timestamp alignments. Specifically, the intersection of two kinds of data timestamps is first obtained, and then partial data with earlier timestamps are removed for the sensor whose timestamp starts earlier. Finally, the timestamps of two sensor signals are synchronized for the following process.

**Author Contributions:** Conceptualization, Q.C. and F.X.; methodology, Q.C.; software, F.X.; validation, Q.C. and F.X.; investigation, H.L.; resources, F.X.; writing—original draft preparation, Q.C. and H.L.; project administration, Q.C.; funding acquisition Q.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by National Key Technology R&D Program of China (No. 2019YFC1606401 & No. 2016YFD0401205), National Natural Science Foundation of China (No. 62076012).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The source code can be seen at https://github.com/xfyy/GaitRecognitionForAndroid.git.

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

## References

1. Ahad, M.A.R.; Ngo, T.T.; Antar, A.D.; Ahmed, M.; Hossain, T.; Muramatsu, D.; Makihara, Y.; Inoue, S.; Yagi, Y. Wearable Sensor-Based Gait Analysis for Age and Gender Estimation. *Sensors* **2020**, *20*, 2424.
2. Jin, M.; He, Y.; Fang, D.; Chen, X.; Meng, X.; Xing, T. iGuard: A real-time anti-theft system for smartphones. *IEEE. Trans. Mob. Comput.* **2018**, *17*, 2307–2320.
3. Terrier, P. Gait Recognition via Deep Learning of the Center-of-Pressure Trajectory. *Appl. Sci.* **2020**, *10*, 774.
4. Prochazka, A.; Dostal, O., Cejnar, P.; Mohamed, I.H.; Pavelek, Z., Vališ, M.; Vyšata, O. Deep Learning for Accelerometric Data Assessment and Ataxic Gait Monitoring. *IEEE Trans. Neur. Sys. Reh. Eng*. **2021**, *29*, 360–367.
5. Guo, C.; Liu, Y.; Song, Qi.; Liu, S. Research on Kinematic Parameters of Multiple Gait Pattern Transitions. *Appl. Sci*. **2021**, *11*, 6911.
6. Limcharoen, P.; Khamsemanan, N.; Nattee, C.; Gait Recognition and Re-Identification Based on Regional LSTM for 2-Second Walks. *IEEE Access* **2021**, 9, 112057–112068.
7. Luo, J.; Tjahjadi, T. Gait Recognition and Understanding Based on Hierarchical Temporal Memory Using 3D Gait Semantic Folding. *Sensors* **2020**, *20*, 1646–1670.
8. Li, C.; Min, X.; Sun, S.; Lin, W.; Tang, Z. DeepGait: A Learning Deep Convolutional Representation for View-Invariant Gait Recognition Using Joint Bayesian. *Appl. Sci.* **2017**, *7*, 210–224.
9. Minvielle, L.; Audiffren, J. NurseNet: Monitoring Elderly Levels of Activity with a Piezoelectric Floor. *Sensors* **2019**, *19*, 3851–3869.
10. Ngo, T.T.; Makihara, Y.; Nagahara, H.; Mukaigawa, Y.; Yagi, Y. Similar gait action recognition using an inertial sensor. *Pattern Recognit.* **2015**, *48*, 1289–1301.
11. Portillo-Portillo, J.; Leyva, R.; Sanchez, V.; Sanchez-Perez, G.; Perez-Meana, H.; Olivares-Mercado, J.; Toscano-Medina, K.; Nakano-Miyatake, M. Cross View Gait Recognition Using Joint-Direct Linear Discriminant Analysis. *Sensors* **2017**, *17*, 6.
12. Lee, S.H.; Lee, S. Fabrication of comb-structured acceleration sensors by roll-to-roll gravure printing. *Int. J. Precis. Eng. Manuf.-Green Technol.* **2021**, *9*, 409–420.

13. Micucci, D.; Mobilio, M.; Napoletano, P. UniMiB SHAR: A Dataset for Human Activity Recognition Using Acceleration Data from Smartphones. *Appl. Sci.* **2017**, *7*, 1101.
14. Tanno, M.; Tanno, H. Aerodynamic characteristics of a free-flight scramjet vehicle in shock tunnel. *Exp. Fluids* **2021**, *62*, 150.
15. Yan, L.; Wei, G.; Hu, Z.; Xiu, H.; Ren, L. Low-cost multisensor integrated system for online walking gait detection. *Sensors* **2021**, *2021*, 6378514.
16. Ding, X.; Wang, K.; Wang, C.; Lan, T.; Liu, L. Sequential convolutional network for behavioral pattern extraction in gait recognition. *Neurocomputing* **2021**, *463*, 411–421.
17. Luo, J.; Wu, H.; Lei, L. GCA-Net: Gait contour automatic segmentation model for video gait recognition. *Multimed. Tools Appl.* **2021**, *4*, 1–13.
18. Gadaleta, M.; Rossi, M. Idnet: Smartphone-based gait recognition with convolutional neural networks. *Pattern Recognit* **2018**, *74*, 25–37.
19. Zou, Q.; Wang, Y.; Wang, Q.; Zhao, Y.; Li, Q. Deep Learning-Based Gait Recognition Using Smartphones in the Wild. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3197–3212.
20. Gao, J.; Gu, P.; Ren, Q.; Zhang, J.; Song, X. Abnormal Gait Recognition Algorithm Based on LSTM-CNN Fusion Network. *IEEE Access* **2019**, *7*, 163180–163190.
21. Alotaibi, M.; Mahmood, A. Improved gait recognition based on specialized deep convolutional neural network. *CVIU* **2017**, *164*, 103–110.
22. Mäntyjärvi, J.; Lindholm, M.; Vildjiounaite, E.; Makela, S.M.; Ailisto, H.A. Identifying users of portable devices from gait pattern with accelerometers. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Philadelphia, Pennsylvania, PA, USA, 18–23 March 2005; pp. 973–976.
23. Derawi, M.O.; Nickel, C.; Bours, P.; Busch, C.C. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In Proceedings of the 2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Darmstadt, Germany, 15–17 October 2010; pp. 306–311.
24. Rong, L.; Zhiguo, D.; Jianzhong, Z.; Ming, L. Identification of Individual Walking Patterns Using Gait Acceleration. In Proceedings of the International Conference on Bioinformatics & Biomedical Engineering, Wuhan, China, 6–8 July 2007; pp. 543–546.
25. Omid, D.; Mojtaba, T.; Raghvendar, C.V. Imu-based gait recognition using convolutional neural networks and multi-sensor fusion. *Sensors* **2017**, *17*, 2735.
26. Fernandez-Lopez, P.; Liu-Jimenez, J.; Kiyokawa, K.; Wu, Y.; Sanchez-Reillo, R. Recurrent Neural Network for Inertial Gait User Recognition in Smartphones. *Sensors* **2019**, *19*, 4054.
27. Hossain Bari,A.,S.,M.;Gavrilova,M.,L. KinectGaitNet: Kinect-Based Gait Recognition Using Deep Convolutional Neural Network. *Sensors* **2022**, *22*, 2631.
28. Huang, H.; Zhou, P.; Li, Y.; Sun,F. A Lightweight Attention-Based CNN Model for Efficient Gait Recognition with Wearable IMU Sensors. *Sensors* **2021**, *21*, 2866.
29. Gill, S.; Seth, N.; Scheme, E. A Multi-Sensor Matched Filter Approach to Robust Segmentation of Assisted Gait. *Sensors* **2018**, *18*, 2970.
30. Sedik, A.; Faragallah, O.S.; El-Sayed, H.S.; El-Banby, G.M.; El-Samie, F.E., Khalaf, A.; El-Shafai, W. An efficient cybersecurity framework for facial video forensics detection based on multimodal deep learning. *Neur. Comp. Appl.* **2022**, *34*, 1251–1268.
31. Wiranto, G.; Kurniawan, D.; Maulana, Y.; Hermida, I.D.P.; Oktaviandi, D. Design and Implementation of Wireless Sensors and Android Based Application for Highly Efficient Aquaculture Management System. *EMITTER Int. J. Eng. Technol.* **2020**, *8*, 355–371.
32. Hwang, S.; Hong, K.; Son, G.; Byun, H. Learning CNN features from DE features for EEG-based emotion recognition. *Pattern Anal. Appl.* **2020**, *23*, 1323–1335.
33. Oluwalade, B.; Neela, S.; Wawira, J.; Adejumo, T.; Purkayastha, S. Human activity recognition using deep learning models on smartphones and smartwatches sensor data. In Proceedings of the 14th International Conference on Health Informatics (HEALTHINF), Vienna, Austria, 11–13 February 2021; pp. 1–6.
34. Yusuf, S.A.; Alshdadi, A.A.; Alassafi, M.O., Alghamdi, R., Samad, A. Predicting catastrophic temperature changes based on past events via a CNN-LSTM regression mechanism. *Neur. Comp. Appl.* **2021**, *33*, 9775–9790.
35. Sugandhi, K.; Wahid, F.F.; Raju, G. Statistical features from frame aggregation and differences for human gait recognition. *Multimed. Tools Appl.* **2021**, *80*, 18345–18364.
36. Francisco, O.; Daniel, R. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* **2016**, *16*, 11501–11525.