Applied Intelligence (2021) 51:4029–4042
https://doi.org/10.1007/s10489-020-02005-7

Deep learning based multimodal complex human activity recognition using wearable devices

Ling Chen[1] · Xiaoze Liu[1] · Liangying Peng[1] · Menghan Wu[1]

Accepted: 5 October 2020 / Published online: 26 November 2020

Abstract
Wearable device based human activity recognition, as an important field of ubiquitous and mobile computing, is drawing more and more attention. Compared with simple human activity (SHA) recognition, complex human activity (CHA) recognition faces more challenges, e.g., various modalities of input and long sequential information. In this paper, we propose a deep learning model named DEBONAIR (Deep lEarning Based multimodal cOmplex humaN Activity Recognition) to address these problems, which is an end-to-end model extracting features systematically. We design specific sub-network architectures for different sensor data and merge the outputs of all sub-networks to extract fusion features. Then, a LSTM network is utilized to learn the sequential information of CHAs. We evaluate the model on two multimodal CHA datasets. The experiment results show that DEBONAIR is significantly better than the state-of-the-art CHA recognition models.

Keywords Complex human activity recognition · Multimodality · Deep learning

1 Introduction

Wearable device based human activity recognition is one of the core research problems of ubiquitous and mobile computing. Human activities can be divided into simple human activities (SHAs) and complex human activities (CHAs). A SHA can be represented as a single repeated action and can be easily recognized by using a single accelerometer. Typical SHAs include "walking", "sitting", and "standing". CHAs are not as repetitive as SHAs and usually involve multiple simultaneous or overlapping actions, which can only be well recognized with multimodal sensor data. Common CHAs include "commuting", "eating", and "house cleaning". Related researches mainly focus on SHAs, which usually describe users' body actions or postures and can be recognized with high accuracy [1–4]. With the growing requirements of many applications (e.g., healthcare systems [5] and smart home [6]), recognizing CHAs begins to attract the attention of the research field.

Existing researches of CHA recognition can be divided into three categories. The first, ignores the differences between CHAs and SHAs, and uses SHA recognition methods to recognize CHAs [7, 8]. The second, represents each CHA by a combination of SHAs, where the SHAs are predefined and labeled manually [9–14]. The last, represents CHAs by latent semantics implied in sensor data, and the latent semantics are discovered by topic models [15–18]. However, they have the following limitations. For the first category, since CHAs are far more complicated than SHAs, the features extracted for SHAs are not representative to CHAs. The second category, heavily relies on domain knowledge, and the predefined SHAs cannot express the components of CHAs precisely, as there are lots of non-semantic and unlabeled components in CHAs. For the third category, topic models only consider distribution information and ignore sequential information, which can contribute to CHA recognition.

Since, sensor data are a kind of time series data, capturing the hidden temporal structure behind them is important for human activity recognition. Hidden Markov Models (HMMs) have been widely used to extract sequential

✉ Ling Chen
lingchen@cs.zju.edu.cn

Xiaoze Liu
liuxz@zju.edu.cn

Liangying Peng
lyoare@zju.edu.cn

Menghan Wu
mfive@zju.edu.cn

[1] College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China

information [19–21]. However, it's difficult for HMMs to capture long-term temporal dependency.

With the development of wearable devices, various kinds of sensor data are used to recognize human activities, and how to fuse these data effectively becomes a challenge. Currently, there are two major fusion approaches: feature level fusion [3, 22] and classifier level fusion [1, 23, 24]. Feature level fusion is to extract features for different sensor data and then concatenate the features to form a new vector. Classifier level fusion is to build base classifiers for different sensor data separately, and then splice the outputs of base classifiers to build a meta-level classifier. However, these two fusion approaches both have some problems. The compatibility problem may occur in the former, as different sensor data have different properties. The latter processes different modalities separately, which considers the properties of different sensor data but cannot extract fusion features.

Recently, deep learning models, e.g., Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have been successfully used in computer vision, natural language processing, speech recognition, etc. Some researchers used these models in human activity recognition and got the state-of-the-art results [24–28]. Unlike traditional methods, deep neural networks organize multilayers to carry out non-linear transformation and linear combination to extract features, which can meet the demand of sequential information extraction and multiple feature fusion. As for human activity recognition, the existing deep neural networks are mostly designed for Inertial Measurement Unit (IMU) based SHA recognition. These networks have not been fully applied to multimodal CHA recognition.

To address these problems, we propose a deep learning model named DEBONAIR (Deep lEarning Based multimodal cOmplex humaN ActivIty Recognition), which has the following characteristics. First, DEBONAIR uses CNNs and Long Short-Term Memory (LSTM) networks to discover sequential information. Second, DEBONAIR introduces specific sub-network architectures, which are designed based on data properties, to handle different sensor data. Third, DEBONAIR fuses multiple sensor data by a convolution layer, which can reserve sequential information and avoid the compatibility problem. We evaluate DEBONAIR on two datasets and compare it with the state-of-the-art models. The experiment results show that DEBONAIR is better than other models.

2 Related work

In this section, we give a review of background work related to human activity recognition, including SHA recognition, CHA recognition, and deep learning based human activity recognition.

2.1 Simple human activity recognition

Early human activity recognition researches mainly concentrated on SHA recognition. Gupta et al. [2] used a single accelerometer placed on the belt clip and classified six SHAs and transitional events. Lara et al. [3] proposed Centinela, which can extract statistical and structural features from acceleration data and vital signs to recognize SHAs. Ravi et al. [4] extracted time and frequency domain features from acceleration data and evaluated a lot of classifiers.

Currently, SHA recognition can achieve high accuracy [2–4]. However, SHA recognition is not sufficient to support some real-world applications, as SHAs (e.g., "sitting" and "walking") are not as reflective as CHAs (e.g., "having a meal" and "shopping").

2.2 Complex human activity recognition

There are three major categories of CHA recognition. The first category, chooses to ignore the differences between CHAs and SHAs, which means the length of samples and the features extracted for CHA recognition are the same as those for SHA recognition. For example, Bao et al. [7] utilized five on-body accelerometers to recognize 20 kinds of activities, containing both SHAs (e.g., "walking" and "standing") and CHAs (e.g., "scrubbing" and "eating or drinking"). Dernbach et al. [29] used acceleration data and orientation data (azimuth, pitch, and roll) to classify SHAs and CHAs in one model. The experiment results demonstrate that the accuracy of CHA recognition is much lower than that of SHA recognition, i.e., the features designed for SHAs are not good at representing CHAs.

The second category, recognizes CHAs by building a hierarchical model, which recognizes SHAs firstly and then represents CHAs by a series of SHAs. Liu et al. [11, 12] built a hierarchical model to recognize CHAs, in which SHAs are predefined by time series patterns, and three shapelet-based models are used to recognize CHAs. Yan et al. [13, 14] designed a 2-tier classification framework, which extracts various features from SHAs to describe CHAs. Although these methods have better performance than methods in the first category, their drawbacks are also obvious. First, predefining SHA labels heavily relays on domain knowledge. Second, the sequences of predefined SHA labels cannot characterize CHAs precisely and effectively, as lots of CHA components are unlabeled and ignored.

The third category, uses latent semantics to model CHAs. Topic models are frequently used in this category to find latent semantics. In these methods, CHAs are treated as "documents", and SHAs are viewed as "words". Huynh et al. [15] recognized CHAs by applying a topic model to SHA sequences. Peng et al. [17, 18] used *k*-means clustering to get the components of CHAs and used a topic model to discover

the latent semantics of CHAs. Topic models only use the distribution of SHAs, while discard lots of sequential information.

2.3 Deep learning based human activity recognition

Nowadays, more and more human activity recognition researchers pay attention to deep learning. Guo et al. [24] used multilayer perceptrons as base classifiers to construct a model, which can utilize unlabeled data to generate diverse base classifiers. Guan et al. [25] integrated various deep learning models to improve human activity recognition accuracy. These models use different initial positions of sampling, different sizes of mini-batches, and different lengths of frames to add the diversity. Ordóñez et al. [26] proposed a deep framework for wearable device based multimodal activity recognition by using convolutional and LSTM recurrent units. Zeng et al. [27] used a CNN to recognize human activities and investigated the optimization of parameters and model architecture. Zheng et al. [28] proposed a multi-channel deep CNN model and evaluated it on two datasets. Yang et al. [30] used a CNN instead of traditional methods (e.g., basis transform coding [31], the statistics of raw signals [32], and symbolic representation [33]) to extract features from row inputs automatically. Münzner et al. [34] investigated several CNN based fusion models for multimodal activity recognition. Chen et al. [35] utilized a CNN plus LSTM framework to recognize activities and users jointly. Yao et al. [36] and Radu et al. [37] adopted sub-networks to process multimodal sensor data, but the same hyper parameters, e.g., the number and size of kernels in a layer, are applied for all sub-networks in each model, which fails to fully consider the different properties of sensor data.

These researchers mainly focused on recognizing SHAs. Unlike SHA recognition, CHA recognition needs longer time windows (referred to as windows below) [38], and the deep neural networks designed for SHA recognition cannot be applied for CHA recognition directly. Although the networks can be adjusted to handle CHA samples, e.g., by increasing the size of convolutional kernels, the tendency within long windows is still hard to be extracted, which can be inferred from the comparison experiments in Section 4.5.2.

Recently, deep learning based CHA recognition has emerged. Peng et al. [39] used a deep multi-task learning framework to recognize CHAs and SHAs jointly, in which a CHA is divided into multiple SHAs. This framework has high demands on datasets, i.e., having both SHA and CHA labels, which greatly limits its applications. In addition, the framework was designed for only a single model (i.e., IMU data). If applying to multimodal data, the framework cannot extract features effectively, as different sensor data have different properties [24], which can be inferred from the comparison experiments in Section 4.5.2. In this paper, we propose a deep learning model for multimodal CHA recognition, which employs specific sub-network architectures for different sensor data.

3 Methodology

Let ca denote a CHA sample, which has an actual label y, where $y \in \{y_i, 1 \le i \le l\}$ and l is the number of CHA labels. The ca is a set of sensor data within a window and can be denoted as $ca = \{TS_k, 1 \le k \le n\}$, where TS_k denotes the time series data measured by the k^{th} sensor, and n denotes the number of sensors. TS_k can be denoted as a $d_k \times n_k$ matrix, where d_k is the dimension number of the sensor (e.g., an accelerometer has three channels: x-axis, y-axis, and z-axis), and n_k is the number of data points within the window. Our task is to build a CHA model f_c to recognize the CHA label y of ca.

The architecture of DEBONAIR is shown in Fig. 1. First, specific sub-network architectures are designed to extract features from different sensor data. Then, the outputs of all sub-networks are merged to extract the latent SHA semantic sequence by the depth concatenation layer and the convolutional layer. After that, an LSTM network is used to get features that take sequential information into consideration. Finally, an output layer that contains a softmax function is used to generate the result.

DEBONAIR contains three components: the convolutional component (including the convolutional sub-networks, the depth concatenation layer, and the convolutional layer), the LSTM layers, and the output layer. In the following subsections, we detail the components of DEBONAIR from bottom to top.

3.1 The convolutional component

In DEBONAIR, CNNs are used to extract features from CHA samples. We first introduce convolutional and pooling layers in CNNs briefly, and then give the design details of the convolutional component.

Convolutional layers Convolutional layers have a set of neurons that filter out the salient patterns of the input. Each neuron contains several trainable kernels to calculate the convolutional results of patterns. The output of each convolutional layer is a set of feature maps, which are computed by applying an activation function to the sum of convolutional results plus the bias.

Pooling layers Pooling operation combines nearby features into a local feature. This operation can also increase the invariance of features and reduce the size of feature maps [40]. We use non-overlapping max pooling to extract enhanced patterns from the previous layer.

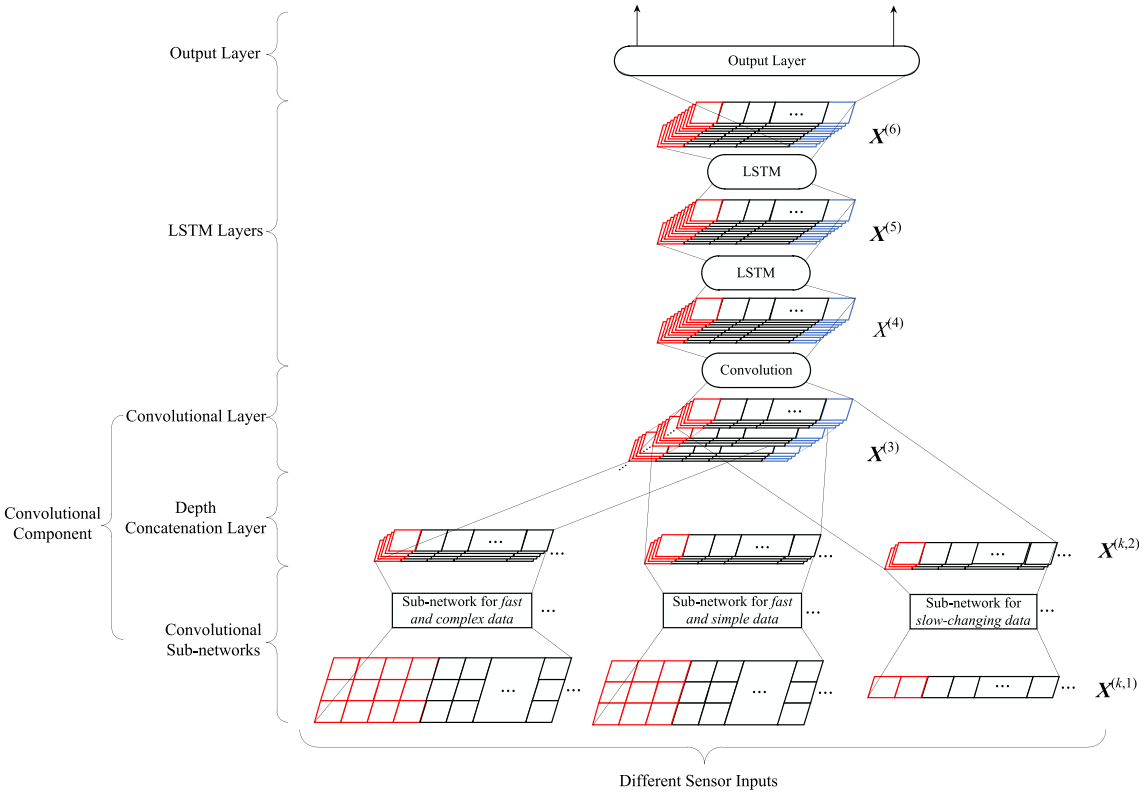

Fig. 1 The architecture of DEBONAIR. $\boldsymbol{X}^{(k,1)}$ denotes the input matrix of the k^{th} sensor, i.e., TS_k. $\boldsymbol{X}^{(k,2)}$ denotes the output matrix of the sub-network corresponding to $\boldsymbol{X}^{(k,1)}$. $\boldsymbol{X}^{(3)}$, $\boldsymbol{X}^{(4)}$, $\boldsymbol{X}^{(5)}$, and $\boldsymbol{X}^{(6)}$ denote the output matrices of the depth concatenation layer, the convolutional layer, the first LSTM layer, and the second LSTM layer, respectively

3.1.1 The convolutional sub-networks

In DEBONAIR, specific sub-network architectures are designed to extract features from different sensor data. We classify the sensor data into three categories according to their properties: 1) *fast and complex data*: data that change fast while the movements behind the data are complex, e.g., data recorded by IMUs placed on wrists; 2) *fast and simple data*: data that change fast and the movements behind the data are simple and almost periodical, e.g., accelerometer data recorded by IMUs placed on legs; 3) *slow-changing data*: data that change slowly and mostly recorded at a low frequency, e.g., heart rate data [41]. To facilitate the description of our model, the frequencies of these three categories are assumed to be 100 Hz, 20 Hz, and 1 Hz, respectively. Note that the sizes of input matrices have a tight connection with the frequencies of sensor data. If the sensor data are not sampled at the assumed frequencies, they can be resampled to match the sub-network architectures.

DEBONAIR adopts three specific sub-network architectures to extract features from different sensor data. The sub-network architectures for IMU data are designed referring to [30]. In order to extract the complex patterns and reduce the effects of noise in the raw data, these two sub-network architectures adopt convolutional kernels with large sizes. Pooling layers with large regions are employed to reduce the dimension number of feature maps rapidly. The sub-network architecture for *fast and complex data* contains three convolutional layers and three pooling layers. Since, the movements behind *fast and simple data* are not as complicated as those of *fast and complex data*, a sub-network architecture with two convolutional layers and two pooling layers is employed. For *slow-changing data*, DEBONAIR employs a sub-network architecture containing consecutive convolutional layers with small kernels and one pooling layer, which is inspired by the networks used in computer vision (e.g., VGGnet [42]). Let $C(n, w)$ denote a convolutional layer with n output filters, and the size of each filter is w. $P(s)$ denotes a non-overlap max-pooling layer, and the size of pooling region is s. The shorthand notations for sub-network architectures that designed for *fast and complex data*, *fast and simple data*, and *slow-changing data* are $C(6, 11) - \mathrm{P}(10) - C(12, 7) - P(5) - C(24, 5) - P(4)$, $\mathrm{C}(6, 11) - \mathrm{P}(10) - \mathrm{C}(12, 5) - \mathrm{P}(4)$, and $C(6, 3) - C(12, 3) - C(12, 3) - P(2)$, respectively.

DEBONAIR uses ReLUs as activation functions in these sub-network architectures. The stride is set to 1 in all convolutional layers. In order to keep the sizes of feature maps unchanged, zero-padding is utilized in all convolutional layers. In order to improve generalization and prevent overfitting, we add a dropout operation after the last layer of

each sub-network architecture, which sets the outputs of randomly-selected neurons to zeros with probability p_{drop} during the training process.

Since, the sub-network architectures do not contain fully-connected layers, the outputs of sub-networks are matrices, i.e., $\{\boldsymbol{X}^{(k,2)}, 1 \le k \le n\}$. Each of them can be denoted as a $\alpha_k \times \beta_k$ matrix, where α_k is the number of feature maps and β_k is the length of feature maps, which is calculated as follows:

$$\beta_k = \frac{f_k \times t}{\prod_i s_i}, \tag{1}$$

where f_k denotes the frequency of the k^{th} sensor data, t denotes the length of windows, and s_i denotes the length of the i^{th} pooling layer. According to the predefined frequencies of sensor data and the architectures of sub-networks, all sub-network outputs $\{\boldsymbol{X}^{(k,2)}, 1 \le k \le n\}$ have the same number of columns, i.e., $t/2$, which is essential to the depth concatenation layer.

3.1.2 The depth concatenation layer and the convolutional layer

A sample contains data sequences from different sensors, DEBONAIR uses the depth concatenation layer and the convolutional layer to extract fusion features, i.e., the latent SHA semantic sequence, which belongs to the category of feature level fusion.

According to the predefined frequencies and the architectures of sub-networks, the outputs of sub-networks have the same number of columns. Thus, as shown in Fig. 1, we apply the depth concatenation operation to the sub-network outputs $\{\boldsymbol{X}^{(k,2)}, 1 \le k \le n\}$ and get $\boldsymbol{X}^{(3)}$, which is the input of the following convolutional layer. The number of columns in $\boldsymbol{X}^{(3)}$ is the same as that in $\boldsymbol{X}^{(k,2)}$, i.e., $t/2$. The number of rows in $\boldsymbol{X}^{(3)}$ is calculated as the sum of the numbers of rows in $\{\boldsymbol{X}^{(k,2)}, 1 \le k \le n\}$.

DEBONAIR employs the convolutional layer to fuse the features extracted from different sensor data, whose kernel size is 1 as suggested in [42, 43]. The number of filters is 50 and the activation function is ReLU. This convolutional layer maps every column in $\boldsymbol{X}^{(3)}$ to the corresponding column in $\boldsymbol{X}^{(4)}$ with a weight matrix, which is shared among all columns. Thus, $\boldsymbol{X}^{(4)}$ can reserve the sequential information by its columns, from which the sequence features can be extracted.

3.2 The LSTM layer

LSTM networks can model sequential information by a memory cell [44], and have been successfully used in various tasks, e.g., activity recognition [26, 34] and video caption [45, 46], we employ a LSTM network to learn the sequential information. We first introduce the architectures of LSTM networks briefly and then depict the LSTM network in DEBONAIR.

A LSTM network [44] is composed of LSTM units, and each unit has a memory cell to remember the important information of time series data. There are three gates, i.e., an input gate, an output gate, and a forget gate, to update, erase, and read out the information in a memory cell, respectively [44]. The gates can determine which part of information (the output of the previous unit and the input data of the current time step) should be remembered or forgotten, which enables LSTM networks to remember the important information even when a sequence is very long.

The workflow of extracting sequence features is shown in Fig. 2. According to previous studies [26, 47, 48], we adopt a two-layer LSTM network to process the latent SHA semantic

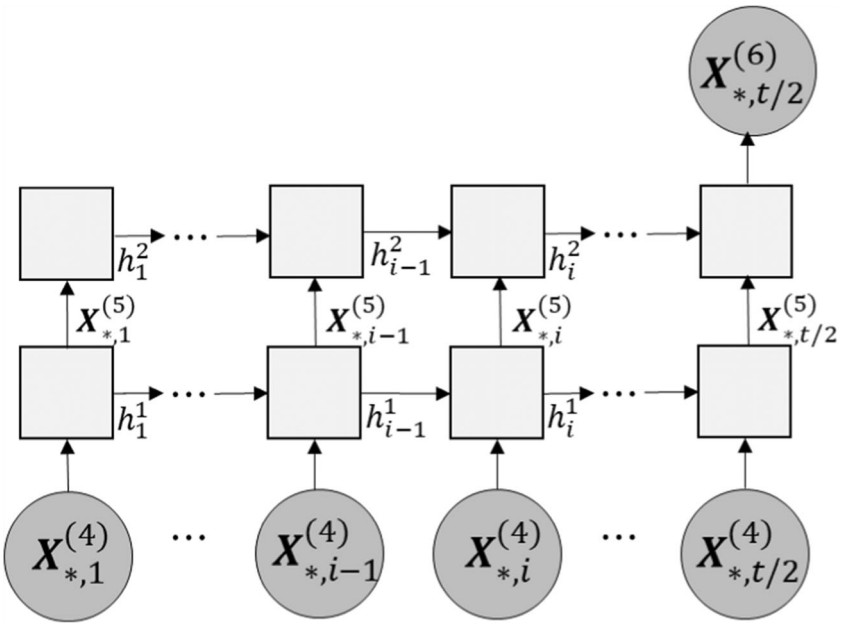

Fig. 2 The workflow of extracting sequence features from the latent SHA semantic sequence by the LSTM network. Circles refer to input and output data. Squares refer to LSTM units. h_i^j refers to the unit's hidden state at the i^{th} time step in the j^{th} layer

sequence. The input of the first LSTM layer is matrix $\boldsymbol{X}^{(4)}$. We feed every column of $\boldsymbol{X}^{(4)}$ to the first LSTM layer in order and get the corresponding column in $\boldsymbol{X}^{(5)}$, which is the input of the next LSTM layer. The output of the second LSTM layer is denoted as $\boldsymbol{X}^{(6)}$. The last column of $\boldsymbol{X}^{(6)}$, i.e., $\boldsymbol{X}^{(6)}_{*,t/2}$, denotes the sequence features.

3.3 The output layer

The output layer contains a softmax function, which takes the sequence features as input and generates the CHA probability distribution. The sequence features $\boldsymbol{X}^{(6)}_{*,t/2}$ is a vector of length n and can be denoted as $\boldsymbol{x}$. The probability of the j^{th} CHA is calculated as follows:

$$P(CHA = j|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{x}}}{\sum_{i=1}^{l} e^{\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x}}} \tag{2}$$

where e refers to the exponential function. $\boldsymbol{w}_i$ $(1 \le i \le l)$ is a trainable parameter vector.

The final CHA label $\widehat{k}$ is set as the one getting the highest probability, i.e., $\widehat{k} = argmax_{j=1}^{l} P(CHA = j|\boldsymbol{x})$.

4 Experiments

In this section, we present the performance evaluation of the proposed model, including datasets, experimental settings, parameter tuning, and experiment results.

4.1 Dataset

In experiments, we utilized two real-world datasets: lifelog dataset and pamap2 dataset. To the best of our knowledge, they are the latest wearable device based multimodal CHA datasets. The statistics of the two datasets are shown in Table 1.

(1) We recruited 4 participants (2 females and 2 males with ages ranging from 21 to 26) to collect lifelog dataset, which contains 80 h of CHA data. Each participant performed 9 kinds of daily CHAs ("commuting", "exercising", "eating", "house cleaning", "meeting", "recreating", "shopping", "sleeping", and "working") by wearing 3 devices: a smartphone, a smartwatch, and a smart chest strap. The smartphone is placed in the front pocket of trousers, the smart watch is placed on the right wrist, and the smart chest strap is tied below axilla. The smartphone and the smartwatch are used to collect acceleration data. The smart chest strap is used to record physiological data, including heart rate, breath rate, heart rate variability, and body posture. All data were labeled by participants at the beginning and the end of the activities. The data were collected in a natural environment that the participants performed complex activities on their own ways without any specific instructions.

(2) Pamap2 dataset was recorded by using three IMUs and one heart rate monitor, including 18 different human activities performed by 9 participants (ages ranging from 23 to 31). Three IMUs are attached on the dominant arm, the chest, and the dominant side's ankle, respectively. Each IMU contains two 3-axis accelerometers, a 3-axis gyroscope, and a 3-axis magnetometer. The data of the lower precision accelerometer in each IMU [41] are discarded. Pamap2 dataset contains over 10 h of activity data. Since, we mainly concentrate on CHA recognition, the activity data with CHA labels (i.e., "Nordic walking", "computer work", "vacuum cleaning", "ironing", "folding laundry", "house cleaning", and "rope jumping") are selected for experiments. Pamap2 dataset is publicly available and can be downloaded from [41]. The data recorded by the

Table 1 The statistics of the utilized datasets

	Lifelog dataset	Pamap2 dataset
# of activities	9	18
# of participants	4	9
Sensors	2 accelerometers and a physiological monitor	3 IMUs and a heart rate monitor
Frequency	Accelerometers: 20 Hz (phone)/100 Hz (watch)	IMUs: 100 Hz
	Physiological monitor: 1 Hz	Heart rate monitor: about 9 Hz
Category	Acceleration (phone): *fast and simple data*	IMU data (hand): *fast and complex data*
	Acceleration (watch): *fast and complex data*	IMU data (ankle): *fast and complex data*
	Physiological data: *slow-changing data*	IMU data (chest): *fast and simple data*
		Heart rate: *slow-changing data*
# of samples	4662	1895

IMU attached on the chest and the heart rate data are resampled to 20 Hz and 1 Hz, respectively.

Due to the unreliability of sensors, there exist missing data in the datasets. The missing values were filled by liner interpolation. In addition, all data were normalized to the range of (0, 1) before inputting into DEBONAIR. The anomalous data were set to boundary values or cut off directly. Then, we segmented the sensor data by sliding windows to generate CHA samples. For lifelog dataset, window length is 60 s and overlap is 50% [18]. For pamap2 dataset, window length is 30 s and overlap is 80% [22].

4.2 Experimental settings

For lifelog dataset, DEBONAIR employs six sub-networks corresponding to acceleration (phone), acceleration (watch), heart rate, breath rate, heart rate variability, and body posture, respectively. Similarly, DEBONAIR employs ten sub-networks for pamap2 dataset, in which each IMU data are processed by three sub-networks, and the rest one is used for heart rate. The categories of the sensor data are given in Table 1.

In the training process, we employ RMSprop [29] as the learning method. Cross-entropy is employed as the loss function. The initial learning rate is 0.01, the batch size is 512, and the dropout probability is 0.1. The parameters used in DEBONAIR are presented in Table 2.

In order to fully investigate the user-independent performance of models, all results are evaluated by Leave-One-Subject-Out cross-validation [49], which uses one participant's data as test data and the other participants' data are randomly split into training set and validation set by a ratio of 0.7: 0.3. This is repeated until all participants' data have been used as test data. The number of training epochs is determined on a per-fold basis using the validation set.

Since the two datasets are class-imbalanced, referring to [26, 50], weighted F1-score is used as the performance metric instead of accuracy, which tends to be higher on more frequent labels.

In order to statistically measure the significance of performance differences, two-tailed paired *t*-tests at 95% significance level are conducted between individual sample predictions of DEBONAIR and each compared model.

4.3 Parameter tuning

In this section, we investigate how the number of training epochs and the LSTM dimension number affect the performance.

Figure 3 shows the classification losses on different numbers of training epochs on lifelog dataset. The training loss and the validation loss gradually decrease with the increasing

Table 2 The parameters used in DEBONAIR

	Layer	Size Per Parameter	Size Per Layer
Layers for *fast and complex data*	Conv1D	**W**: 6 × 11 × 3 **b**: 6	204
	Conv1D	**W**: 12 × 7 × 6 **b**: 12	516
	Conv1D	**W**: 24 × 5 × 12 **b**: 24	1464
Layers for *fast and simple data*	Conv1D	**W**: 6 × 11 × 3 **b**: 6	204
	Conv1D	**W**: 12 × 5 × 6 **b**: 12	372
Layers for *slow-changing data*	Conv1D	**W**: 6 × 3 × 1 **b**: 6	24
	Conv1D	**W**: 24 × 3 × 6 **b**: 24	456
Layers after concatenation	Conv1D	**W**: 50 × 1 × 204 **b**: 50	10,250
	PReLU	15 × 50	750
	LSTM	$\mathbf{W}_i$, $\mathbf{W}_f$, $\mathbf{W}_c$, $\mathbf{W}_o$: 100 × 50 $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_c$, $\mathbf{b}_o$: 1 × 50	20,200
	LSTM	$\mathbf{W}_i$, $\mathbf{W}_f$, $\mathbf{W}_c$, $\mathbf{W}_o$: 100 × 50 $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_c$, $\mathbf{b}_o$: 1 × 50	20,200
	FC	**W**: 7 × 750 **b**: 7	5257

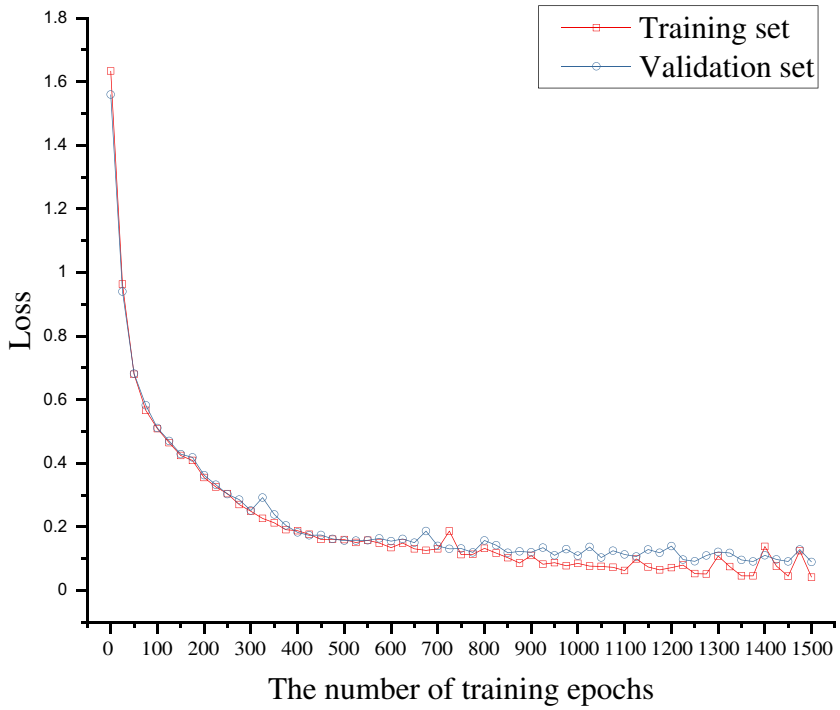

Fig. 3 The impact of the number of training epochs

number of training epochs. When the number of epochs increases to 500, the validation loss is relatively stable and the network is close to converging. After 750 epochs, the gap between the two losses is gradually increased. Therefore, the number of training epochs is set in the range of [500, 750]. After 500 epochs, the training process stops if the validation loss does not decrease.

The LSTM dimension number is vital to our model. We gradually increase the LSTM dimension number from 10 to 90 with an increment step of 20 and measure model performance on lifelog dataset. Results are shown in Fig. 4, which indicate that when the LSTM dimension number increases, the performance of DEBONAIR first increases and then decreases. A possible reason is that when the LSTM dimension number is too small, the information memorized by the LSTM network is not enough to represent CHAs. When the LSTM dimension number is too large, the LSTM network contains excessive number of parameters, which may reduce the generalization ability of the model. Thus, the LSTM dimension number is set to 50 in the following experiments.

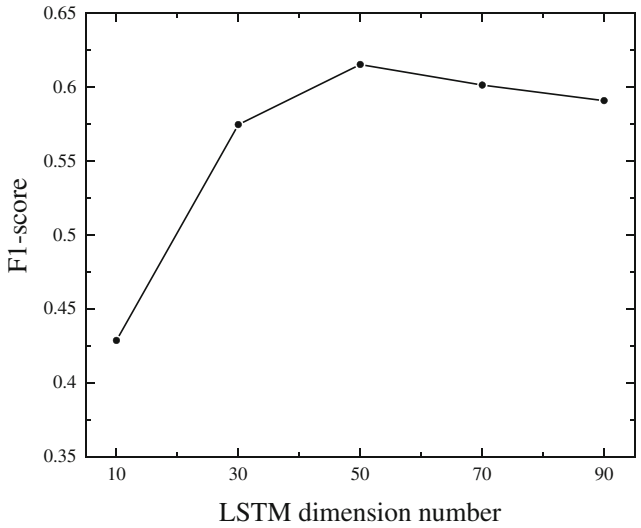

Fig. 4 The impact of LSTM dimension number

4.4 The experiment results of DEBONAIR

In order to analyze the effectiveness of DEBONAIR, we train it based on the aforementioned settings. Figure 5 shows the confusion matrices of DEBONAIR on the two datasets. The value in the i^{th} row and the j^{th} column denotes the ratio of classifying CHA samples with the i^{th} category as the j^{th} category.

For lifelog dataset, DEBONAIR has worse performance on classifying “meeting”, “eating”, and “recreating”, which tend to be misclassified as “working”. It is obvious that these CHAs are based on SHA “sitting”. In addition, DEBONAIR tends to misclassify “commuting” as “house cleaning”. A possible reason is that these two CHAs usually contain same SHAs, e.g., “walking”, which make them difficult to be distinguished.

For pamap2 dataset, DEBONAIR performs worse on “folding laundry” and “house cleaning” than other CHAs. CHA “folding laundry” tends to be misclassified as “ironing” and “house cleaning”, as all these CHAs contain similar SHAs, e.g., “standing still and moving hands”. CHA “house

	1	2	3	4	5	6	7	8	9
1	48.24%	10.21%	2.56%	21.87%	1.45%	2.78%	5.12%	0.82%	6.94%
2	12.50%	78.29%	0.45%	0.35%	0.00%	0.74%	7.41%	0.16%	0.10%
3	4.98%	1.07%	43.43%	0.42%	3.75%	4.22%	0.67%	0.12%	41.35%
4	14.06%	1.90%	1.35%	66.47%	2.33%	2.08%	9.48%	0.42%	1.90%
5	0.14%	0.00%	6.49%	0.04%	60.97%	0.00%	0.00%	0.00%	32.37%
6	0.15%	0.00%	6.60%	0.34%	0.10%	34.51%	0.76%	3.20%	54.34%
7	14.81%	1.92%	1.83%	4.81%	0.07%	15.01%	58.87%	0.73%	1.94%
8	0.02%	0.00%	1.11%	0.00%	0.01%	6.07%	0.01%	85.38%	7.39%
9	0.18%	0.00%	8.11%	0.39%	6.62%	12.52%	0.05%	0.92%	71.21%

(a) Lifelog dataset. 1=commuting, 2=exercising, 3=eating, 4=house cleaning, 5=meeting, 6=recreating, 7=shopping, 8=sleeping, 9=working

	1	2	3	4	5	6	7
1	98.51%	0.00%	0.76%	0.00%	0.00%	0.43%	0.31%
2	0.00%	90.55%	1.10%	3.21%	1.41%	3.73%	0.00%
3	1.56%	0.49%	89.51%	1.02%	0.49%	6.93%	0.00%
4	0.00%	3.30%	2.17%	80.35%	9.13%	5.05%	0.00%
5	0.00%	2.01%	5.95%	29.44%	40.89%	21.70%	0.00%
6	1.45%	6.25%	20.05%	8.30%	12.99%	49.36%	1.61%
7	6.10%	0.00%	0.11%	0.00%	0.00%	1.58%	92.20%

(b) Pampa2 dataset. 1=Nordic walking, 2=computer working, 3=vacuuming, 4=ironing, 5=folding laundry, 6=house cleaning, 7=rope jumping

Fig. 5 The confusion matrices of DEBONAIR. (a) Lifelog dataset. 1 = commuting, 2 = exercising, 3 = eating, 4 = house cleaning, 5 = meeting, 6 = recreating, 7 = shopping, 8 = sleeping, 9 = working. (b) Pampa2 dataset. 1 = Nordic walking, 2 = computer working, 3 = vacuuming, 4 = ironing, 5 = folding laundry, 6 = house cleaning, 7 = rope jumping

cleaning" also tends to be misclassified as "vacuuming". It might be because that "house cleaning" contains SHA "moving objects and putting them back again", which is also the component of "vacuuming", i.e., "moving chairs during vacuuming".

4.5 Comparison with other models

In order to demonstrate the effectiveness of DEBONAIR, we compare it with the simplified models and the state-of-the-art models.

4.5.1 Comparison with the simplified models

In order to evaluate the effectiveness of DEBONAIR's architecture, we specifically design seven simplified models, which are DEBONAIR-1 Hz, DEBONAIR-20 Hz, DEBONAIR-100 Hz, DEBONAIR-NO-LSTM, DEBONAIR-NO-CONV, DEBONAIR-LSTM-TO-FC, and DEBONAIR-LSTM-TO-CONV. In the first three simplified models, the architectures of all sub-networks are the same. All sensor data are sampled to 1 Hz/20 Hz/100 Hz to match the sub-network architecture designed for slow-changing data/fast and simple data/fast and complex data, respectively. DEBONAIR-NO-LSTM removes the LSTM network from DEBONAIR. DEBONAIR-NO-CONV removes the convolutional layer after the depth concatenation layer in the convolutional component. DEBONAIR-LSTM-TO-FC replaces the LSTM layers with two fully connected (FC) layers, having 64 and 32 neurons, respectively. DEBONAIR-LSTM-TO-CONV replaces the LSTM layers with two convolutional layers, each having 64 filters with size 6.

The classification performance of these models is given in Table 3, and the following tendencies could be discerned:

(1) The performance of DEBONAIR is better than that of DEBONAIR-1 Hz, DEBONAIR-20 Hz, and DEBONAIR-100 Hz, which demonstrates the effectiveness of the specific designed sub-network architectures. Since different categories of data have different properties, the features of them cannot be well extracted by sub-networks of a same architecture.
(2) The performance of DEBONAIR is better than that of DEBONAIR-NO-CONV, which demonstrates that the convolutional layer can improve the performance of the model by reducing the dimension number of feature maps and extracting fusion features.
(3) Compared with DEBONAIR-NO-LSTM, DEBONAIR-LSTM-TO-FC, and DEBONAIR-LSTM-TO-CONV, DEBONAIR has better performance on both datasets.

Table 3 The F1-scores of DEBONAIR and simplified models (mean ±std). * indicates that DEBONAIR is statistically superior to the compared model (pairwise *t*-test at 95% significance level)

	Lifelog	Pamap2
DEBONAIR	**0.615 ± 0.133**	**0.836 ± 0.163**
DEBONAIR -1 Hz	0.580 ± 0.143*	0.721 ± 0.222*
DEBONAIR -20 Hz	0.564 ± 0.132*	0.772 ± 0.173*
DEBONAIR -100HZ	0.573 ± 0.128*	0.769 ± 0.181*
DEBONAIR -NO-LSTM	0.560 ± 0.123*	0.794 ± 0.185*
DEBONAIR -NO-CONV	0.565 ± 0.139*	0.777 ± 0.206*
DEBONAIR -LSTM-TO-FC	0.593 ± 0.136*	0.806 ± 0.194*
DEBONAIR -LSTM-TO-CONV	0.573 ± 0.154*	0.813 ± 0.202*

This indicates that the LSTM network can learn sequential information effectively.

4.5.2 Comparison with other models

To show the competitive performance of DEBONAIR, we compare it with the following models.

- **Hierarchy**: Hierarchy [18] is a topic model based method. For acceleration data, it divides each acceleration sample into finer-grained segments and calculates some statistical features (as shown in Table 4) for all segments. Then these segments are clustered by the *k*-means clustering algorithm, and each cluster denotes a component of CHAs. After that, LDA topic model is used to discover the latent semantics of CHAs, and a base classifier is built on it. For physiological data, Hierarchy extracts structural and transient features and builds a base classifier directly. Finally, two base classifiers are fused by a meta classifier to get the final CHA labels. The base classifiers and the meta classifier are J48 and multinomial logistic regression, respectively.
- **Non-hierarchy**: Non-hierarchy [3] is a traditional method designed for SHA recognition. It extracts statistics features from acceleration data, as well as structural and transient features from physiological data (as shown in Table 4). Then all these features are concatenated into a new feature vector. Finally, a decision tree classifier is built on the new feature vector.
- **Hybrid-LSTM**: Hybrid-LSTM modifies Hierarchy by employing a LSTM network instead of LDA topic model. For acceleration data, it obtains the components of CHAs in the same way as Hierarchy. Then Hybrid-LSTM gets the embedding vectors of all components and applies a LSTM network on them. For physiological data, Hybrid-LSTM extracts physiological features in the same way as Hierarchy. At last, the features of acceleration and physiological data are concatenated into a feature vector, and softmax is employed as the final classifier.
- **DeepConvLSTM**: DeepConvLSTM [26] is a deep learning based SHA recognition model, which contains four convolutional layers and two LSTM layers. In order to apply DeepConvLSTM to CHA recognition, we rescale the length of convolutional kernels according to the size of CHA samples.
- **CB-LF**: CB-LF [34] is a channel-based late fusion model, which comprises four convolutional layers, a fully-connected layer, and two LSTM layers. Different from DeepConvLSTM, in this model, each sensor axis is treated as an individual channel and processed by using different convolutional kernels. Then the outputs are fused through the fully-connected layer.
- **DeepSense**: DeepSense [36] is a deep learning based multimodal SHA recognition model, which contains three individual convolutional layers and three merge convolutional layers. DeepSense also adopts two GRU layers to learn temporal features. To apply DeepSense, all sensor data from lifelog dataset are resampled to 1 Hz and sensor data from pamap2 dataset are resampled to about 9 Hz.
- **SADeepSense**: SADeepSense [51] is the state-of-the-art deep learning based multimodal SHA recognition model, which integrates two self-attention modules (sensor attention module and temporal attention module) with DeepSense to merge information from multiple sensors and over time. Specifically, two transformation functions

Table 4 The features used in Hierarchy, Non-hierarchy, and Hybrid-LSTM

Statistical features	Mean, variance, standard deviation, mean absolute deviation, root mean square, Pearson's linear correlation coefficient, energy, interquartile rage
Structural features	The polynomials of degree one, two, three: $f = a_0 + a_1 t$; $f = a_2 + a_3 t + a_4 t^2$; $f = a_5 + a_6 t + a_7 t^2 + a_8 t^3$
Transient features	The trend (increasing, decreasing, or constant), the magnitude of change

Table 5 The parameters of Hybrid-LSTM, DeepConvLSTM, CB-LF, DeepSense, and SADeepSense. The numbers before and after "@" refer to the number and size of kernels, respectively. The parameters of the LSTM/GRU layer and embedding vector are dimension numbers

	Convolutional layer		LSTM/GRU layer		Embedding vector	
	Lifelog	Pamap2	Lifelog	Pamap2	Lifelog	Pamap2
Hybrid-LSTM	–	–	20	20	10	10
DeepConvLSTM	64@250 for all layers	64@125 for all layers	128	128	–	–
CB-LF	64@250 for all layers	64@125 for all layers	128	128	–	–
DeepSense	64@2 for all layers	64@2 for all layers	120	120	–	–
SADeepSense	64@2 for all individual convolutional layers 64@8, 64@6, 64@4 for merge convolutional layers	64@2 for all individual convolutional layers 64@8, 64@6, 64@4 for merge convolutional layers	120	120	–	–
	9@1, 9@6 for sensor attention module	7@1, 7@10 for sensor attention module				
	9@1, 9@5 for temporal attention module	7@1, 7@9 for temporal attention module				

f and *g* are exploited to extract local and global correlation features in each self-attention module, which are implemented by two convolutional layers.

The parameters of Hierarchy are the same as [18]. The parameters of Hybrid-LSTM, DeepConvLSTM, CB-LF, DeepSense, and SADeepSense are optimized and given in Table 5.

The classification performance of these models is given in Table 6, and the following tendencies could be discerned:

1 The classification performance of Hierarchy is poorer than that of DEBONAIR. It not only justifies the benefits of deep specific sub-network architectures, but also shows the effect of the LSTM network. Sub-networks of different architectures can extract features from different sensor data. In addition, LSTM networks are better at learning sequential information than topic models.

2 The performance of Hybrid-LSTM is poorer than that of DEBONAIR. Since the difference between these two models is that Hybrid-LSTM employs traditional features while DEBONAIR utilizes deep features, the comparison result shows that the features learnt by the convolutional component are more representative than traditional features.

3 Non-hierarchy, DeepConvLSTM, CB-LF, DeepSense, and SADeepSense perform worse than DEBONAIR. It suggests that it is hard to achieve satisfactory CHA recognition performance when applying an SHA recognition model directly. DeepConvLSTM, CB-LF, DeepSense, and SADeepSense are all deep learning based models that contain CNN and RNN networks, but they exploited convolutional layers with the same architecture to extract features from different sensor data, which fails to fully consider the difference of sensor data. The performance of DeepConvLSTM is better than that of CB-LF, it might be because that CB-LF processes each channel differently,

Table 6 The numbers of parameters, F1-scores, and computation time of DEBONAIR and all compared models (mean±std).

	Lifelog			Pamap2		
	# of parameters	F1-score	computation time (ms)	# of parameters	F1-score	computation time (ms)
DEBONAIR	52,153	**0.615 ± 0.133**	0.098	71,585	**0.836 ± 0.163**	0.241
Hierarchy	–	0.427 ± 0.043*	10.406	–	0.501 ± 0.152*	13.032
Non-hierarchy	–	0.396 ± 0.091*	0.156	–	0.682 ± 0.163*	0.041
Hybrid-LSTM	14,157	0.447 ± 0.082*	0.105	62,031	0.672 ± 0.174*	0.209
DeepConv-LSTM	3,463,817	0.346 ± 0.048*	1.243	1,991,559	0.746 ± 0.141*	0.317
CB-LF	33,198,089	0.328 ± 0.031*	11.453	49,715,463	0.701 ± 0.103*	7.171
DeepSense	521,281	0.557 ± 0.025*	0.212	1,041,839	0.766 ± 0.041*	0.529
SADeepSense	519,547	0.563 ± 0.152*	0.236	1,005,841	0.832 ± 0.143*	0.611

*indicates that DEBONAIR is statistically superior to the compared model (pairwise *t*-test at 95% significance level)

which increases model's complexity and makes CB-LF prune to overfitting.

4 Hybrid-LSTM performs better than Hierarchy. It indicates the importance of sequential information. The topic model employed by Hierarchy only uses the distribution of SHAs and discards the sequential information, which can be learnt by LSTM networks.

5 Compared to the pure deep learning based models, i.e., DeepConvLSTM, CB-LF, DeepSense, and SADeepSense, DEBONAIR performs better while has fewer parameters and takes shorter computation time (average training time), which justifies its efficiency.

5 Conclusions and future work

CHA recognition is a kernel problem in ubiquitous and mobile computing. In this paper, we propose DEBONAIR, an attempt to unitize deep neural networks to extract features from different sensor data for CHA recognition. The experimental results justify the effectiveness of designing specific sub-network architectures for different types of sensor data, which is instructive to the CHA recognition field. However, the experimental results shown in Section 4.4 present DEBONAIR's relatively low performance on differentiating CHAs containing same or similar SHAs, indicating that DEBONAIR cannot fully model the complex temporal structure hidden in human activities, which remains to be further studied.

In addition, we will extend our work in the following directions. Firstly, we will employ attention mechanism in our model to make full use of all the states of the LSTM network. Secondly, we will use more information (e.g., location context) to further improve the performance.

Acknowledgments This work is supported by the Fundamental Research Funds for the Central Universities (No. 2020QNA5017).

References

1. Guo H, Chen L, Shen Y, Chen G (2014) Activity recognition exploiting classifier level fusion of acceleration and physiological signals. In *Proceedings of the 16th International Joint Conference on Pervasive and Ubiquitous Computing*, 63–66
2. Gupta P, Dallas T (2014) Feature selection and activity recognition system using a single triaxial accelerometer. IEEE Trans Biomed Eng 61(6):1780–1786
3. Lara OD, Pérez AJ, Labrador MA, Posada JD (2012) Centinela: a human activity recognition system based on acceleration and vital sign data. Pervasive and Mobile Computing 8(5):717–729
4. Ravi N, Dandekar N, Mysore P, Littman ML (2005) Activity recognition from accelerometer data. In *Proceedings of the 20th Association for the Advance of Artificial Intelligence*, 1541–1546
5. Taraldsen K, Chastin SF, Riphagen II, Vereijken B, Helbostad JL (2012) Physical activity monitoring by use of accelerometer-based body-worn sensors in older adults: a systematic literature review of current knowledge and applications. Maturitas 7(1):13–19
6. Chen L, Nugent CD, Wang H (2012) A knowledge-driven approach to activity recognition in smart homes. IEEE Trans Knowl Data Eng 24(6):961–974
7. Bao L, Intille SS (2004) Activity recognition from user-annotated acceleration data. In *Proceedings of the 2nd International Conference on Pervasive Computing*, 1–17
8. Dernbach S, Das B, Krishnan NC, Thomas BL, Cook DJ (2012) Simple and complex activity recognition through smart phones. In *Proceedings of the 8th International Conference on Intelligent Environments*, 214–221
9. Banke U, Schiele B (2009) Daily routine recognition through activity spotting. In *Proceedings of the 4th International Symposium on Location-and Context-Awareness*, 192–206
10. Blanke U, Schiele B (2010) Remember and transfer what you have learned-recognizing composite activities based on activity spotting. In *Proceeding of the 14th International Symposium on Wearable Computers*, 1–8
11. Liu L, Peng Y, Liu M, Huang Z (2015) Sensor-based human activity recognition system with a multilayered model using time series shapelets. Knowl-Based Syst 90:138–152
12. Liu L, Peng Y, Wang S, Liu M, Huang Z (2016) Complex activity recognition using time series pattern dictionary learned from ubiquitous sensors. Inf Sci 340:41–57
13. Yan Z, Chakraborty D, Misra A, Jeung H, Aberer K (2012) Sammple: Detecting semantic indoor activities in practical settings using locomotive signatures. In *Proceeding of 16th International Symposium on Wearable Computers*, 37–40
14. Yan Z, Chakraborty D, Mittal S, Misra A, Aberer K (2013) An exploration with online complex activity recognition using cellphone accelerometer. In *Proceeding of the 15th International Conference on Pervasive and Ubiquitous Computing*, 199–202
15. Huynh T, Fritz M, Schiele B (2008) Discovery of activity patterns using topic models. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, 10–19
16. Lv M, Chen L, Chen T, Chen G (2018) Bi-view semi-supervised learning based semantic human activity recognition using accelerometers. IEEE Trans Mob Comput 17(9):1991–2001
17. Peng L, Chen L, Wu M, Chen G (2019) Complex activity recognition using acceleration, vital sign, and location data. IEEE Trans Mob Comput 18(7):1488–1498
18. Peng L, Chen L, Wu X, Guo H, Chen G (2017) Hierarchical complex activity representation and recognition using topic model and classifier level fusion. IEEE Trans Biomed Eng 64(6):1369–1379
19. Trabelsi D, Mohammed S, Chamroukhi F, Oukhellou L, Amirat Y (2013) An unsupervised approach for automatic activity recognition based on hidden Markov model regression. IEEE Trans Autom Sci Eng 10(3):829–835
20. Guan X, Raich R, Wong WK (2016) Efficient multi-instance learning for activity recognition from time series data using an auto-regressive hidden Markov model. In *Proceedings of the 33rd International Conference on Machine Learning*, 2330–2339
21. Martindale CF, Sprager S, Eskofier BM (2019) Hidden Markov model-based smart annotation for benchmark cyclic activity recognition database using wearables. Sensors 19(8):1820
22. Tapia EM, Intille SS, Haskell W, Larson K, Wright J, King A, Friedman R (2007) Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor. In *Proceedings of the 11th International Symposium on Wearable Computers*, 37–40
23. Guo M, Wang Z, Yang N, Li Z, An T (2019) A multisensor multiclassifier hierarchical fusion model based on entropy weight for human activity recognition using wearable inertial sensors. IEEE Transactions on Human-Machine Systems 49(1):105–111

24. Guo H, Chen L, Peng L, Chen G (2016) Wearable sensor based multimodal human activity recognition exploiting the diversity of classifier ensemble. In *Proceedings of the 18th International Joint Conference on Pervasive and Ubiquitous Computing*, 1112–1123
25. Guan Y, Plötz T (2017) Ensembles of deep LSTM learners for activity recognition using wearables. In *Proceedings of the 19th ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(2): 11
26. Ordóñez FJ, Roggen D (2016) Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. Sensors 16(1):115
27. Zeng M, Nguyen LT, Yu B, Mengshoel OJ, Zhu J, Wu P, Zhang J (2014) Convolutional neural networks for human activity recognition using mobile sensors. In *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services*, 197–205
28. Zheng Y, Liu Q, Chen E, Ge Y, Zhao JL (2014) Time series classification using multi-channels deep convolutional neural networks. In *Proceedings of the 15th International Conference on Web-Age Information Management*, 298–310
29. Dauphin Y, Vries H, Bengio Y (2015) Equilibrated adaptive learning rates for non-convex optimization. In *Proceedings of the 29th Advances in Neural Information Processing System*, 1504–1512
30. Yang J, Nguyen MN, San PP, Li X, Krishnaswamy S (2015) Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 25–31
31. Huynh T, Schiele B (2005) Analyzing features for activity recognition. In *Proceedings of the 1st Joint Conference on Smart Objects and Ambient intelligence: Innovative Context-aware Services: Usages and Technologies*, 159–163
32. Bulling A, Blanke U, Schiele B (2014) A tutorial on human activity recognition using body-worn inertial sensors. ACM Comput Surv 46(3):33
33. Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series, with implications for streaming algorithms. *In Proceedings of the 8th Special Interest Group on Management of Data*, 2–11
34. Münzner S, Schmidt P, Reiss A, Hanselmann M, Stiefelhagen R, Dürichen R (2017) CNN-based sensor fusion techniques for multimodal human activity recognition. In *Proceedings of the 21st ACM International Symposium on Wearable Computers*, 158–165
35. Chen L, Zhang Y, Peng L (2020) METIER: a deep multi-task learning based activity and user recognition model using wearable sensors. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1): 1–18
36. Yao S, Hu S Zhao Y, Zhang A, Abdelzaher T (2017) Deepsense: a unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the International Conference on World Wide Web*, 351–360
37. Radu V, Tong C, Bhattacharya S, Lane ND, Mascolo C, Marina MK, Kawsar F (2018) Multimodal deep learning for activity and context recognition. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4): 157
38. Shoaib M, Bosch S, Incel OD, Scholten H, Havinga PJ (2016) Complex human activity recognition using smartphone and wrist-worn motion sensors. Sensors 16(4):426
39. Peng L, Chen L, Ye Z, Zhang Y (2018) AROMA: A deep multi-task learning based simple and complex human activity recognition method using wearable sensors. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(2): 74
40. Boureau YL, Ponce J, LeCun Y (2010) A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning*, 111–118
41. Reiss A, Stricker D (2012) Introducing a new benchmarked dataset for activity monitoring. In *Proceedings of the 16th International Symposium on Wearable Computers*, 108–109
42. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations*, 1–14
43. Lin M, Chen Q, Yan S (2014) Network in network. In *Proceedings of the 2nd International Conference on Learning Representations*, 1–10
44. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
45. Gao L, Li X, Song J, Shen HT (2019) Hierarchical LSTMs with adaptive attention for visual captioning. IEEE Trans Pattern Anal Mach Intell 42(5):1112–1131
46. Song J, Guo Y, Gao L, Li X, Hanjalic A, Shen HT (2019) From deterministic to generative: multimodal stochastic RNNs for video captioning. IEEE Transactions on Neural Networks and Learning Systems 30(10):3047–3058
47. Andrej K, Justin J, Li FF (2016) Visualizing and understanding recurrent networks. *In Proceedings of the 4th International Conference on Learning Representations*, 1–12
48. Sainath TN, Vinyals O, Senior A, Sak H (2015) Convolutional, long short-term memory, fully connected deep neural networks. In *Proceeding of the 40th International Conference on Acoustics, Speech and Signal Processing*, 4580–4584
49. Hammerla NY, Plötz T (2015) Let's (not) stick together: Pairwise similarity biases cross-validation in activity recognition. In *Proceedings of the 17th International Joint Conference on Pervasive and Ubiquitous Computing*, 1041–1051
50. Hammerla NY, Halloran S, Plötz T (2016) Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 1533–1540
51. Yao S, Zhao Y, Shao H, Liu D, Liu S, Hao Y, Piao A, Hu S, Lu S, Abdelzaher T (2019) SADeepSense: Self-attention deep learning framework for heterogeneous on-device sensors in internet of things applications. In *Proceedings of the 38th IEEE Conference on Computer Communications*, 1243–1251

Ling Chen received the B.S. and Ph.D. degrees in computer science from Zhejiang University, China, in 1999 and 2004, respectively. He is currently an associate professor with the College of Computer Science and Technology, Zhejiang University, China. His research interests include ubiquitous computing, AI, pattern recognition, and human computer interaction.

Xiaoze Liu received the B.S. degree in computer science and technology from Xidian University, China, in 2015 and the M.S. degree in computer science and technology from Zhejiang University, China, in 2018. His research interests include wearable computing and human activity recognition.

Menghan Wu received the B.S. and M.S. degrees in computer science and technology from Zhejiang University, China, in 2016 and 2019, respectively. Her research interests include wearable computing and human activity recognition.

Liangying Peng received the B.S. degree in software engineering from Nankai University, China, in 2014 and the Ph.D. degree in computer science and technology from Zhejiang University, China, in 2020. Her research interests include wearable computing and human activity recognition.