AROMA: A Deep Multi-Task Learning Based Simple and Complex Human Activity Recognition Method Using Wearable Sensors

LIANGYING PENG, Zhejiang University, China
LING CHEN[*], Zhejiang University, China
ZHENAN YE, Zhejiang University, China
YI ZHANG, Zhejiang University, China

Human activity recognition (HAR) is a promising research issue in ubiquitous and wearable computing. However, there are some problems existing in traditional methods: 1) They treat HAR as a single label classification task, and ignore the information from other related tasks, which is helpful for the original task. 2) They need to predesign features artificially, which are heuristic and not tightly related to HAR task. To address these problems, we propose AROMA (human activity recognition using deep multi-task learning). Human activities can be divided into simple and complex activities. They are closely linked. Simple and complex activity recognitions are two related tasks in AROMA. For simple activity recognition task, AROMA utilizes a convolutional neural network (CNN) to extract deep features, which are task dependent and non-handcrafted. For complex activity recognition task, AROMA applies a long short-term memory (LSTM) network to learn the temporal context of activity data. In addition, there is a shared structure between the two tasks, and the object functions of these two tasks are optimized jointly. We evaluate AROMA on two public datasets, and the experimental results show that AROMA is able to yield a competitive performance in both simple and complex activity recognitions.

CCS Concepts: • **Human centered computing → Ubiquitous and mobile computing**; *Ubiquitous and mobile computing design and evaluation methods*;

Additional Key Words and Phrases: Human activity recognition, multi-task learning, deep learning, LSTM

ACM Reference Format:

Liangying Peng, Ling Chen, Zhenan Ye, and Yi Zhang. 2018. AROMA: A Deep Multi-Task Learning based Simple and Complex Human Activity Recognition Method Using Wearable Sensors. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 2, Article 74 (June 2018), 16 pages. https://doi.org/10.1145/3214277

1 INTRODUCTION

Human activity recognition (HAR) is one of the most significant and valuable issues in ubiquitous and wearable computing, as it can support many real-word applications, e.g., healthcare [5], skills assessment [32], and industrial assistant [21]. Although there are a variety of methods that can recognize human activities with high

[*] This is the corresponding author

Authors' addresses: L. Peng, College of Computer Science and Technology, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China, lyoare@zju.edu.cn; L. Chen, College of Computer Science and Technology, Alibaba-Zhejiang University Joint Institute of Frontier Technologies, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China, lingchen@zju.edu.cn; Z. Ye, College of Computer Science and Technology, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China, zhenanye@zju.edu.cn; Y. Zhang, College of Computer Science and Technology, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China, zhangyi1995@zju.edu.cn.

2474-9567/2018/6-ART74 $15.00
https://doi.org/10.1145/3214277

accuracy, they mainly focus on simple activities, e.g., “running”, “sitting”, and “walking”. Simple activities are usually characterized by repeated actions [6] or single body posture [14]. Compared with simple activities, complex activities are more complicated, and have higher level semantics, e.g., “working”, “having dinner”, and “commuting” [25]. They are more reflective of people’s daily lives.

Most of the prior work treated HAR as a single classification task. Given a sequence of activity data, a HAR model generally extracts some predesigned features from the activity data, and then maps the features to an activity label. These features are usually designed manually, and the most common ones are statistical features [2] and structural features [16]. However, these features are heuristic and not task-dependent [33], which have been widely used in many time series problems. Nowadays, some researchers have tried to apply deep learning methods to discover deep features for HAR [8, 10, 24, 33, 34]. Deep learning is a class of machine learning algorithms that are based on learning data representations [33]. Common deep learning methods, e.g., convolutional neural networks (CNNs) [17], recurrent neural networks (RNNs) [28], and long short-term memory (LSTM) networks [13], are good at discovering deep features automatically, which can effectively characterize the nature of raw data. CNNs are feedforward neural networks, compared with CNNs, RNNs add a directed cycle to exhibit dynamic temporal behaviors, i.e., RNNs have “memory” to learn temporal relationships on time series data. LSTM networks extend RNNs with more complex memory cells and can effectively avoid the long-term dependency problem of RNNs [7].

These deep learning based HAR methods still focused only on a single classification task. In this way, information coming from related tasks is ignored, which might help HAR models do better [27]. Multi-task learning is a technique that can solve multiple tasks jointly. It is a promising way to improve model performance by exploiting the commonalities and differences across multiple tasks. There is a shared structure among multiple tasks. If the tasks are chosen appropriately, the shared structure can help each task learn more from other tasks. There is limited work on multi-task learning based HAR. Sun et al. [29-30] proposed a multi-task learning method for personalized HAR, where each task corresponds to a specific person. However, this work still relied on traditional statistical features.

To address the above problems, we propose AROMA (human activity recognition using deep multi-task learning) to solve simple and complex activity recognition tasks jointly. In HAR, simple and complex activity recognitions are a pair of related tasks. On one hand, simple activities act as the components of complex activities [25]. For example, when a user is performing complex activity “commuting”, there are many possible simple activities, e.g., “walking”, “standing still”, and “going upstairs”. On the other hand, the features of complex activities can reflect the relationships of simple activities, and further help simple activity recognition. When recognizing simple and complex activities, multi-task learning can discover the commonalities between these two kinds of activities, which are hard to learn by single task learning. Meanwhile, the differences between simple and complex activities are a kind of inductive bias, which can help improve the generalization abilities of both simple and complex activity recognition models. Specifically, there is a shared structure between these two tasks, whose parameters are optimized jointly.

The main contributions of this work are summarized as follows:

1) Propose AROMA, a deep multi-task learning based method to solve simple and complex activity recognition tasks jointly, which can exploit the commonalities and differences between these two tasks to achieve better recognition performance and generalization ability.

2) Design a CNN for simple activity recognition task, which can extract deep features that are relative to HAR task and non-handcrafted. In addition, we combine a deep residual network with the CNN to achieve shallower network and better recognition performance.

3) Apply a LSTM network for complex activity recognition task, which can utilize the temporal context of activity data. There is a shared structure between these two tasks, in which the input of the LSTM network is the output of the CNN.

4) Evaluate AROMA on two public datasets and perform comparison with other methods. The experimental results show that AROMA is able to yield a competitive performance in both simple and complex activity recognitions.

The remainder of this paper is organized as follows: Section 2 gives a survey of the related work. Then Section 3 describes the proposed method AROMA. Section 4 presents the experimental results. Finally, Section 5 concludes the paper.

2 RELATED WORK

This section presents the related work in the area of wearable sensor based HAR, including HAR (simple activity recognition and complex activity recognition), as well as deep learning and multi-task learning for HAR.

2.1 Human Activity Recognition

Most current HAR studies mainly involve in recognizing simple activities. For instance, Gupta et al. [9] used a waist-mounted triaxial accelerometer to recognize simple activities, e.g., "sitting", "walking", and "running". Lara et al. [16] used a smartphone and a chest strap to recognize simple activities, e.g., "walking", "ascending", "descending", and "sitting". He et al. [12] recognized four human activities (i.e., "running", "still", "jumping", and "walking") by using a single triaxial accelerometer.

Compared with simple activity recognition, complex activity recognition has developed much slower. Existing studies on complex activity recognition can be classified into two categories. The first one does not distinguish complex activities from simple activities, and uses methods designed for simple activities to recognize complex activities [2, 6]. For example, Dernbach et al. [6] used a triaxial accelerometer and a gyroscope to recognize simple activities (e.g., "standing" and "sitting") as well as complex activities (e.g., "cleaning kitchen" and "cooking"). Bao et al. [2] utilized five biaxial accelerometers to recognize twenty human activities, ranging from simple activities (e.g., "walking" and "running") to complex activities (e.g., "working" and "eating"), which demonstrated that the recognition performance of complex activities is far lower than that of simple activities. Since the structure of complex activities is more complicated, the methods designed for simple activity recognition are weak in classifying complex activities.

The second one defines complex activities in a hierarchical way, where a complex activity is represented by a combination of simple activities [3, 4, 14, 18, 19, 25,]. For example, Huynh et al. [14] used a hierarchical model to recognize activity patterns as a probabilistic combination of simple activities, where the simple activities are manually labeled. Liu et al. [18, 19] proposed a multilayered model to represent complex activities and employed a shapelet-based framework to recognize them. Blanke et al. [3-4] proposed a top-down method, which identifies simple activities and reveals them to classify complex activities through a feature selection algorithm. However, these methods heavily rely on domain knowledge. They not only have to predefine simple activities, but also have to manually design features from simple activity sequences to characterize complex activities. Since there are a large number of fine-grained components in complex activities, which may not be represented by predefined simple activities, these methods would lose effective and symbolic information of classification. To address this problem, Peng et al. [25] generated simple activity labels by clustering, and discovered the latent semantics of complex activities based on a topic model. But topic models define a document (i.e., a complex activity) based on a bag of words (i.e., components of complex activities) model, which ignores word order. That is to say, the temporal context of activity data is lost.

In addition, all the above methods treated HAR as a single label classification task. They did not consider the correlation between simple and complex activities, which could help improve the performance of their HAR models. In this work, we regard simple and complex activity recognitions as two related tasks. They are learnt jointly, and what is learned for one task can help the other learn better.

2.2 Deep Learning for HAR

Deep learning is a kind of machine learning algorithm that uses multiple layers of nonlinear processing units to learn data representations. It contains multiple layers, where each successive layer uses the output from the previous layer as input. Deep learning methods have been successfully applied in many fields, e.g., computer vision, speech recognition, and natural language processing. Some researchers have also tried to apply deep learning methods for HAR [8, 10, 24, 33, 34]. For instance, Zeng et al. [34] utilized a CNN to extract discriminative features for HAR, which could capture the local dependency and scale invariance of activity data. Yang et al. [33] adopted a deep CNN to extract features automatically from time series data to recognize human activities and hand gestures. Ordóñez and Roggen [24] used deep convolutional and LSTM recurrent neural networks for wearable sensor based multimodal HAR. Their work demonstrated that combining CNN and LSTM network could achieve a higher recognition accuracy than only using a CNN, which inspired our work. Guan and Plötz [8] built various LSTM networks and fused them in score-level to improve HAR performance. Hammerla et al. [10] investigated the impact of several different deep learning methods (e.g., deep feed-forward network, CNN, and bi-directional LSTM network) on HAR.

All these studies demonstrated the effectiveness of deep learning for HAR. Meanwhile, they also faced a problem that deeper networks were able to achieve better performance, but were harder to get convergence. To address this problem, we utilize a deep residual network in AROMA, which is shallow and easy to be trained. More importantly, it can achieve the same or even better performance than deeper networks.

2.3 Multi-task Learning for HAR

Multi-task learning is a technique that multiple tasks are solved jointly using a shared structure. If the tasks are chosen well, the shared structure contributes to improving the generalization ability of the model. There are two kinds of multi-task learning algorithms. The first one is homogeneous multi-task learning, in which each task corresponds to a single output. For example, digit recognition can use multi-task learning methods by casting it as 10 binary classification tasks [15]. The second one is heterogeneous multi-task learning that each task corresponds to a unique set of outputs. For example, given a face picture, one task is to predict the person's age and the other is to identify the gender [35].

Multi-task learning has been used successfully in many applications, e.g., speech recognition, computer vision, and drug discovery. There is limited work on multi-task learning based HAR using wearable sensors. Sun et al. [29-30] proposed a personalized HAR method by applying multi-task learning, where each task corresponds to a specific person. However, this method still relied on traditional statistical features, and each task in it had the same structure for HAR. In this work, we build a homogeneous multi-task learning architecture and meet a greater challenge, i.e., the structures for different tasks are very different.

3 METHODOLOGY

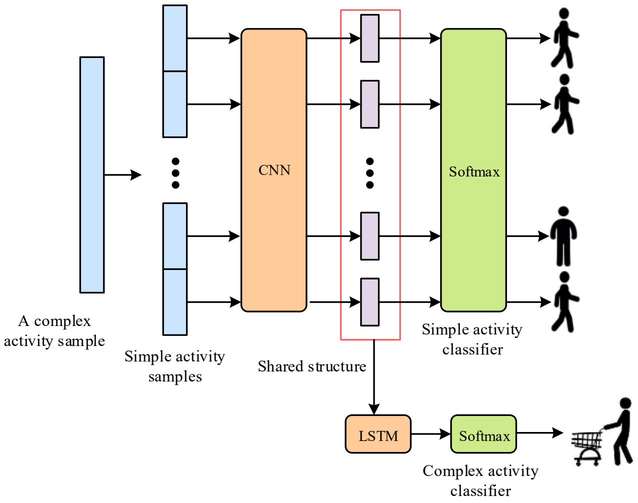
A complex activity sample
Simple activity samples
CNN
Shared structure
Softmax
Simple activity classifier
LSTM
Softmax
Complex activity classifier

Fig. 1. The architecture of AROMA.

This section gives a detailed description of our method AROMA, which recognizes simple and complex activities jointly. AROMA divides raw sensor data to fixed length windows, and each complex activity sample contains multiple simple activity samples. As shown in Fig. 1, the biggest blue rectangle refers to a complex activity sample, which is divided into multiple segments as simple activity samples (i.e., other blue rectangles). For simple activity recognition task, AROMA utilizes a CNN to extract deep features (i.e., the purple rectangles), which are then input into a softmax classifier to predict simple activity labels. For complex activity recognition task, the deep features after the CNN are concatenated together forming a deep feature layer (i.e., the red rectangle), and then input into a LSTM network. After that, a complex activity classifier is trained to predict complex activity labels. Simple and complex activity recognitions are two tasks in this work, and they have a shared structure. Intuitively, simple activities are the components of a complex activity, thus, we regard the CNN and the deep feature layer as the shared structure.

This work is inspired by other related studies [23, 24], both applied a CNN plus LSTM structure for HAR. Softmax is the most commonly used classifier in deep learning methods to represent a probability distribution. Thus, we apply softmax as the classifiers for both simple and complex activities, which are trained jointly in AROMA.

3.1 Problem Definition

The input data utilized in this work are naturally indexed over time dimension, allowing us to segment them into fixed length windows. Although, there is not a unified system to define simple and complex activities, we give the definitions of them according to [4, 6, 20].

Definition 1 (Window) A window is a three-tuple $win = (X, t_s, t_e)$, in which X denotes the input data, t_s and t_e are the starting and ending time of the window, respectively.

Definition 2 (Simple activity sample) Simple activities are low-level human activities characterized by body actions or posture [6, 20], e.g., “running”, “walking”, and “sitting”. A simple activity sample is defined as a window with fixed length ls, which could be recognized as a simple activity category. Simple activity samples are usually obtained by using a fixed length window to segment the raw sensor data [20].

Definition 3 (Complex activity sample) Complex activities are high-level human activities with semantic meaning of daily lives [4, 6, 20], e.g., “working”, “having dinner”, and “shopping”. Similar with simple activity sample, a complex activity sample is defined as a window with fixed length lc, which could be recognized as a complex activity category. Since complex activities are more complicated and non-homogeneous [20], their window length should be longer than that of simple activities to capture their features, i.e., $lc > ls$.

Definition 4 (Simple activity recognition task) Given a simple activity sample sa and its simple activity label ys, the goal is to find a mapping function $f_s: sa \rightarrow ys$, and predicted label $f_s(sa)$ should be as similar as possible to the actual label.

Definition 5 (Complex activity recognition task) Similar with simple activity recognition task, complex activity recognition task is to find a mapping function $f_c: ca \rightarrow yc$, where ca is a complex activity sample labeled with yc. Predicted label $f_c(ca)$ should be as similar as possible to the actual label.

Definition 6 (Simple and complex activity recognition joint task) Given a set of activity data, and its corresponding simple and complex activity labels, train a simple activity classifier f_s and a complex activity classifier f_c by minimizing total loss $L(f_s) + L(f_c)$, where $L(f_s)$ and $L(f_c)$ denote the classification losses of f_s and f_c, respectively.

3.2 Simple Activity Recognition Task

AROMA uses a CNN to discover deep features to represent simple activity samples. We first introduce CNN and then explain how to use it in AROMA.

A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers are either convolutional, pooling, or fully connected. The input is composed by activity data obtained from wearable sensors. Data are processed layer by layer, where each layer uses the output of the previous layer as input. Each layer contains multiple units, and we use u_i^l to denote the ith unit in layer l. In addition, x_i^l denotes the value of u_i^l.

Convolutional layer. Convolutional layers act as feature extractors to provide abstract representations of the input data. In convolutional layers, the previous layer's output is convolved with several convolutional kernels to yield feature maps. Formally, extracting a feature map using a convolution operation is given by:

$$a_j^{l+1} = \sigma\left(\sum_{f=1}^{F^l} K_{j,f}^l a_f^l + b_j^l\right) \tag{1}$$

where a_j^l represents the jth feature map in layer l, $\sigma()$ is a non-linear activation function, F^l is the number of feature maps in layer l, $K_{j,f}^l$ is the kernel convolved over feature map f in layer l to create feature map j in layer $(l+1)$, and b_j^l is a bias vector.

Pooling layer. Pooling is a form of non-linear down-sampling. Pooling layers serve to progressively reduce the spatial size of representations, and reduce the number of parameters. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN. We use max pooling to map the output of the previous layer:

$$x_i^{l+1} = max_{k=1}^{r}(x_{i+k}^l) \tag{2}$$

where x_i^l is the value of the ith unit in layer l, and r is the length of the pooling region.

Fully connected layer. Finally, after several convolutional and pooling layers, the high-level reasoning of a neural network is done via fully connected layers, which unify the output of feature maps to achieve a parametric concatenation by:

$$\boldsymbol{z}^{l+1} = x_i^{l}\circ\boldsymbol{w} \tag{3}$$

where vector $\boldsymbol{z}$ contains unnormalized log probabilities, and $\boldsymbol{w}$ is a weight vector.

Generally, in order to get better performance for a deep neural network model, researchers often try to add more layers. However, deeper networks are not able to achieve expected performance, and might lead to higher training error [11]. To address this problem, He et al. [11] proposed a deep residual learning framework, and applied it for image recognition. A deep residual network consists of multiple residual units. Under the premise of ensuring that the dimension of a residual unit's input is the same as that of convolutional layers' outputs, the input of the residual unit and the outputs of multiple convolutional layers are summed together and then activated by a rectified linear unit (ReLU) function. In this way, the output of the residual unit can be obtained.

In traditional CNNs, both convolutional and pooling layers can serve as down-sampling functions to reduce the dimension of data, i.e., each layer has lossy compression effects and there is the problem of information loss. By applying a residual unit, the output of the first layer is skipped directly to the input of the third layer, thus, the "clear" data of the first layer and the lossy compressed data of the second layer are put together as the input of the third layer, which can reduce the information loss. Compared with adding a large number of layers to networks, deep residual networks are shallower and easier to be trained.

We utilize residual learning in AROMA. Fig. 2 describes the workflow of simple activity recognition task. The CNN contains eight convolutional layers, two pooling layers, a fully connected layer, and four residual units. In experiments, we compared different numbers of convolutional layers, and found that this network (i.e., eight convolutional layers) can achieve the best performance.

Generally, the activity data from wearable sensors are multi-channel data. Channel represents the dimension of data. For instance, triaxial accelerations have three channels. In this work, we use a matrix $X_{\alpha\times\beta}$ to denote the activity data of a simple activity sample, where α is the number of channels, β is the length of data. Correspondingly, AROMA employs 2D convolutional kernels, whose size is the number before @ in Fig. 2. All the activity data are normalized firstly and then input to the CNN. For each layer (except the first layer) in the CNN, the number of feature maps is the number of kernels (i.e., the number after @ in Fig. 2) in the previous layer. For example, the number of feature maps in the second layer is the number of kernels in the first layer, i.e., 32. Note that, when using a CNN for image processing, the input data (e.g., an image) have a depth of 3 (i.e., R, G, and B). But in this work, the depth of activity data is 1, thus, we set the number of feature maps at the input level $F^1 = 1$.

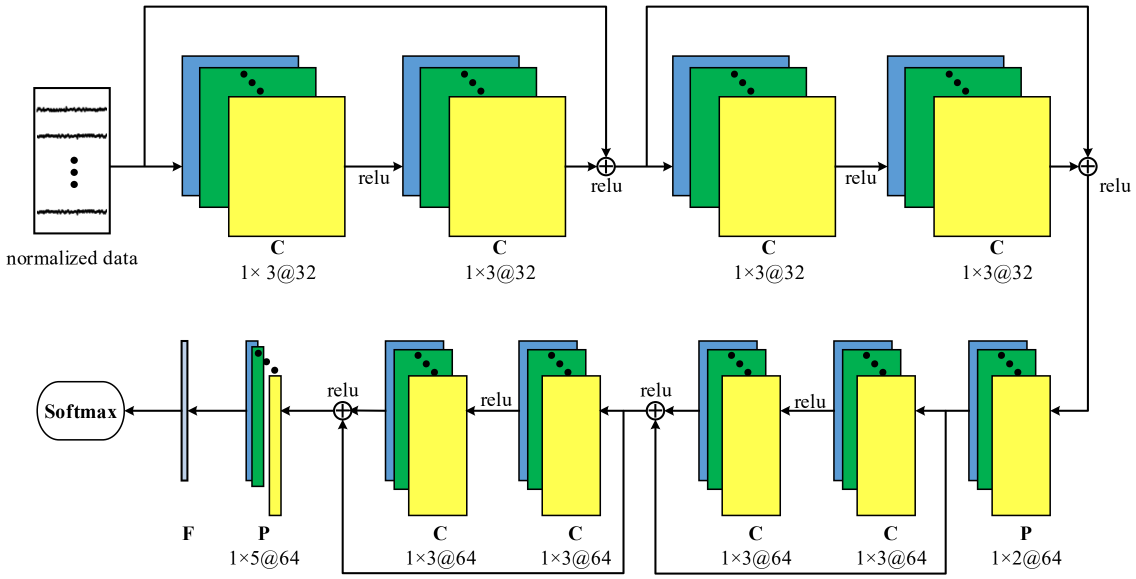

Fig. 2. The workflow of simple activity recognition task. Symbols "C", "P", and "F" refer to convolutional, pooling, and fully connected layers, respectively. The numbers before @ refer to the parameters of convolutional kernels or pooling operations. The numbers after @ mean the number of convolutional kernels.

In convolutional layers, ReLUs are employed to compute feature maps. After CNN, the output of the last fully connected layer is served as the input of a softmax classifier, which predicts simple activity labels, i.e., the output of the last layer is governed by a softmax function:

$$f_s(x) = P(ys = al|x) = \frac{e^{z_{ys_{al}}}}{\sum_{j=1}^{n} e^{z_j}} \tag{4}$$

where al is a simple activity label, ys is the output of simple activity classifier $f_s(x)$, n is the number of simple activity labels, and z_j means the jth element of unnormalized log probability vector $\boldsymbol{z}$. The predicted simple activity label is assigned to the one with the highest probability, i.e., $al \leftarrow argmax_{al=1}^{n} P(ys = al|x)$. Other parameters defining the CNN are presented in Fig. 2.

3.3 Complex Activity Recognition Task

AROMA applies a LSTM network to recognize complex activities. LSTM networks are an extension of RNNs. A RNN is an artificial network where connections between units form a directed cycle, which allows information to flow both forward and backward within the network. RNNs can use their memory cells to exhibit dynamic temporal behaviours and learn the temporal context of input data.

A LSTM network is composed of LSTM units, and each unit has a memory cell $\boldsymbol{c}_t$, which can be updated, erased, and read out [13]. There are three kinds of "gates" that control the reading, writing, and memory updating: input gate $\boldsymbol{i}_t$, output gate $\boldsymbol{o}_t$, and forget gate $\boldsymbol{f}_t$. Compared with RNNs, LSTM networks are better at learning temporal relationships on long time scales.

For an exemplary one-layer network, a LSTM unit takes input data $\boldsymbol{x}_t$, hidden state $\boldsymbol{h}_{t-1}$, and cell state $\boldsymbol{c}_{t-1}$ from the previous LSTM unit. The hidden state is updated at every time step t. Given the input, the output of a LSTM unit can be written as follows:

$$\boldsymbol{i}_t = sigm(W_{xi}\boldsymbol{x}_t + W_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{5}$$

$$\boldsymbol{f}_t = sigm(W_{xf}\boldsymbol{x}_t + W_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \tag{6}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ tanh(W_{xc}\boldsymbol{x}_t + W_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \tag{7}$$

$$\boldsymbol{o}_t = sigm(W_{xo}\boldsymbol{x}_t + W_{ho}\boldsymbol{h}_{t-1} + W_{co}\boldsymbol{c}_{t-1} + \boldsymbol{b}_o) \tag{8}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \circ tanh(\boldsymbol{c}_t) \tag{9}$$

where $\boldsymbol{c}_t$ and $\boldsymbol{h}_t$ are the outputs of the LSTM unit, and can be passed to the next time step to iterate the aforementioned process. Operator ° stands for element-wise multiplication. W is a weight matrix, with subscripts representing from-to relationship. For instance, W_{xi} is the input-input gate matrix, and W_{hf} is the hidden-forget gate matrix. Variables $\boldsymbol{b}_i$, $\boldsymbol{b}_f$, $\boldsymbol{b}_c$, and $\boldsymbol{b}_o$ are bias vectors.

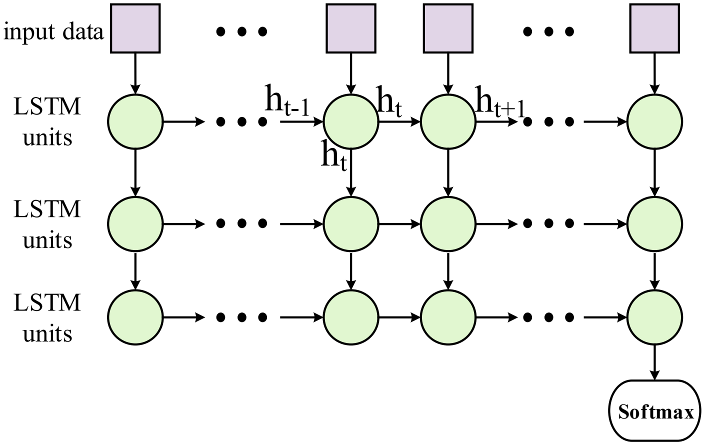

Fig. 3. The workflow of complex activity recognition task. The green circles are LSTM units.

Fig. 3 shows the workflow of complex activity recognition task. We adopt a three-layer LSTM network according to [10]. As we know, within a single complex activity sample, there are lc/ls simple activity samples. Thus, each layer contains lc/ls LSTM units. The output of the previous LSTM unit is input to the next unit. In this way, the temporal context of activity data can be learnt. The input of the first layer is the output of the fully connected layer of the CNN, which is a shared structure between simple and complex activity recognition tasks. The output of the LSTM network is served as the input of a softmax classifier, which predicts complex activity labels. That is to say, given $\boldsymbol{h}_t$, prediction can be performed and the complex activity probability distribution vector $\boldsymbol{pc} = [pc_1, pc_2, ..., pc_m]$ is calculated as follows:

$$\boldsymbol{pc} = s({W_{hm}}^T \boldsymbol{h}_t + \boldsymbol{b}_m) \tag{10}$$

where m is the number of complex activity labels, W_{hm} and $\boldsymbol{b}_m$ are the weight and bias for the output layer (i.e., classification layer), respectively. $s()$ is a softmax function (see Equation (4)). Similar with simple activity recognition task, the predicted complex activity label is assigned to the one with the highest probability, i.e., $al \leftarrow argmax_{al=1}^{m} pc_{al}$.

3.4 Multi-task Learning

Let $f_s(sa)$ and $f_c(ca)$ denote two classifiers on simple activity sample sa and complex activity sample ca, respectively. Both simple and complex activity recognition tasks use softmax (Equation (4)) as classifiers. The loss functions of these classifiers are calculated based on their negative log-likelihood:

$$L(f_s) = -log f_c(sa) \tag{11}$$

$$L(f_c) = -log f_c(ca) \tag{12}$$

The goal of AROMA is to train two satisfactory classifiers by minimizing these two loss functions as follows:

$$f_s, f_c \leftarrow argmin\, L(f_s) + L(f_c) \tag{13}$$

When training AROMA, there is a backpropagation process that the error from recognizing complex activities can go back to simple activity classifier, which is described in Algorithm 1.

Algorithm 1 The updating process of shared weight parameter

```
Input:    shared weight parameter: $\boldsymbol{w}$.
          learning rate: $\theta$.
          the number of iterations: $\lambda$.
Output:   updated shared weight parameter $\nabla\boldsymbol{w}$.
1.        initialize AROMA as given or default network structures
2.        Repeat
3.            #Forward Propagation
4.            calculate $L(f_s)$ using Equation (11)
5.            calculate $L(f_c)$ using Equation (12)
6.            calculate total loss $L_{total} = L(f_s) + L(f_c)$
7.            #Backward Propagation
8.            $\nabla\boldsymbol{w} \leftarrow \theta \frac{\partial L(f_s)}{\partial \boldsymbol{w}} + \theta \frac{\partial L(f_c)}{\partial \boldsymbol{w}}$
9.        until converge or reach the number of iterations $\lambda$
10.       return updated shared weight $\nabla\boldsymbol{w}$
```

For each iteration, there are two kinds of steps, i.e., forward propagation (lines 3-6) and backward propagation (lines 7-8). When we minimize the total loss of simple and complex activity classifiers, the backpropagation process will affect the weight parameter of the shared structure.

4 EXPERIMENTS

In this section, we introduce the experiments for evaluating the proposed method AROMA. Firstly, we describe datasets and experimental setup. Second, we investigate the impact of parameters (e.g., the number of convolutional layers) on classification performance. Then, we give the convergence processes and confusion matrices of AROMA. Finally, we compare our method with several state-of-the-art HAR methods.

4.1 Experimental Dataset

We use two public HAR datasets in our experiments: Ubicomp 08 dataset [14] and Opportunity dataset [26]. To the best of our knowledge, they are the most widely used and latest available wearable sensor based HAR datasets with two-level activity labels (e.g., simple and complex activity labels). The details of two datasets are given in Table 1.

(1) Ubicomp 08 dataset [14] contains two-level activity labels: activities and daily routines, where activities are the components of daily routines. Thus, they correspond to simple and complex activities, respectively. The dataset contains 84 hours (in seven days) of activity data from only one participant. The participant wore two devices, both of which collect triaxial accelerations. One was worn on the right wrist, and the other was put into a trousers' pocket. The dataset does not provide raw data but two statistical features (i.e., mean and variance) on triaxial accelerations. There are around 15000 simple activity samples and 1000 complex activity samples in the original dataset, but some of them are unlabeled, which are not used in this work. This dataset is publicly available and can be downloaded from [31].

(2) Opportunity dataset [26] consists of around 6 hours of activity data from 4 participants. The participants performed activities of daily living (ADL) with various sensors of different modalities, including 5 commercial inertial measurement units (IMU), 2 commercial inertial sensors, and 12 Bluetooth acceleration sensors. Each IMU is composed of a 3D accelerometer, a 3D gyroscope, and a 3D magnetic sensor. Each participant performed 5 ADL sessions and a drill session. During each ADL session, participants performed activities with no constrain. During the drill session, participants performed 20 repetitions of a predefined sorted set of 17 activities. Data were annotated at several layers, high-level activities, medium-layer activities (i.e., arm movements), low-level activities (i.e., left and right arm movements and object usage), and locomotion. In this work, we regard high-level activities and locomotion as complex and simple activities, respectively. Unlabeled samples are not used in this work. This dataset is publicly available and can be downloaded from [22].

Table 1. The details of datasets. The numbers in the parentheses mean the number of activity samples. The forward slashes mean "or".

	Ubicomp 08	Opportunity
# of participants	1	4
Sensors	2 triaxial accelerometer	5 IMUs, 2 inertial sensors, and 12 acceleration sensors
Frequency	2.5Hz	30Hz
Simple activities	sitting/ desk activities (9233), lying while reading/ using computer (604), having dinner (383), walking freely (380), driving car (368), having lunch (229), discussing at whiteboard (191), attending a presentation (149), driving bike (141), watching a movie (129), standing/ talking on phone (75), walking while carrying something (70), walking (69), picking up cafeteria food (69), sitting/ having a coffee (66), queuing in line (60), personal hygiene (52), using the toilet (51), fanning barbecue (46), washing dishes (39), kneeling/ doing something else (35), sitting/ talking on phone (26), kneeling/ making fire for barbecue (25), setting the table (24), standing/ having a coffee (20), preparing food (14), having breakfast (14), brushing teeth (13), standing/ using the toilet (9), standing/ talking (8), washing hands (6), making coffee (5), running (3), wiping the whiteboard (2)	stand (10330), walk (6041), sit (6545), lie (1338)
Complex activities	dinner activities (44), commuting (58), lunch routine (79), office work (579)	relaxing (661), coffee time (1578), early morning (1447), cleanup (1304), sandwich time (3083)

4.2 Experimental Setting

In the step of dividing raw data into fixed length samples, we set simple activity sample length ls to 20 seconds and complex activity sample length lc to 5minutes for Ubicomp 08 dataset. Note that the sample lengths of both simple and complex activities are shorter compared to [14], as we apply deep learning methods that require a large number of training samples, and shorter sample length can bring more samples. For opportunity dataset, ls and lc are set to 500ms and 10s referring to [24].

In the iteration process of AROMA, the learning rate θ in Algorithm 1 is dynamically optimized, which is usually set from 0.0007 to 0.0001.

Since Ubicomp 08 dataset was collected from only one person, we apply leave-one-day-out cross validation referring to [14], which leaves one day's data as test data and the remaining six days' data are training data. This is repeated until all days' activity data have been used as test data. For Opportunity dataset, following [24], AROMA is trained on data of all ADL and drill sessions of participant 1 and ADL 1, ADL 2, ADL 3, and drill sessions of participants 2 and 3. Data of ADL 4 and ADL 5 sessions of participants 2 and 3 are used as test data. As shown in Table 1, two datasets are class-imbalanced. As stated in [25], accuracy usually evaluates the performance of a classifier in regard of frequent classes, it often leads to false positive classification while keeping high accuracy. Therefore, referring to [1], we take micro-averaged F-score (abbreviated as F-score) as the performance measure of our experiments:

$$F - score = \frac{2\sum_{cn=1}^{CN} TP_{cn}}{2\sum_{cn=1}^{CN} TP_{cn} + \sum_{cn=1}^{CN} FP_{cn} + \sum_{cn=1}^{CN} FN_{cn}} \tag{14}$$

where CN denotes class number. Variables TP_{cn}, FP_{cn}, and FN_{cn} are the true positives, false positives, and false negatives of class cn, respectively. F-score is an informative evaluation metric that involves precision and recall. When conducting cross-validation, F-score summarizes all true positives, false positives, and false negatives over all classes and all folds to evaluate a classifier's performance [1].

4.3 Impact of Parameters

To obtain a reliable architecture of AROMA, we evaluate the influence of the number of convolutional layers. We tune this parameter with the total loss of simple and complex activity classifiers on Ubicomp 08 dataset. We compare four structures, whose parameters and performances are given in Table 2 and Fig. 4. It can be found that increasing the number of convolutional layers tends to improve the F-score of both complex and simple activity recognitions. Thus, we employ eight convolutional layers in the CNN of AROMA.

Table 2. The parameters of compared four CNN structures. Symbols "C", "P", and "F" refer to convolutional, pooling, and fully connected layers, respectively. The numbers before @ refer to the parameters of convolutional kernels or pooling operations. The numbers after @ mean the number of convolutional kernels.

#of convolutional layers	Parameters of CNN
2	C (1×3@64)→C (1×3@64)→P (1×5@64)→F
4	C (1×3@32)→C (1×3@32)→P (1×2@64)→C (1×3@64)→C (1×3@64)→P (1×5@64)→F
6	C (1×3@32)→C (1×3@32)→C (1×3@32)→C (1×3@32)→P (1×2@64)→ C (1×3@64)→C (1×3@64)→P (1×5@64)→F
8	C (1×3@32)→C (1×3@32)→C (1×3@32)→C (1×3@32)→P (1×2@64)→ C (1×3@64)→C (1×3@64)→ C (1×3@64)→C (1×3@64)→P (1×5@64)→F

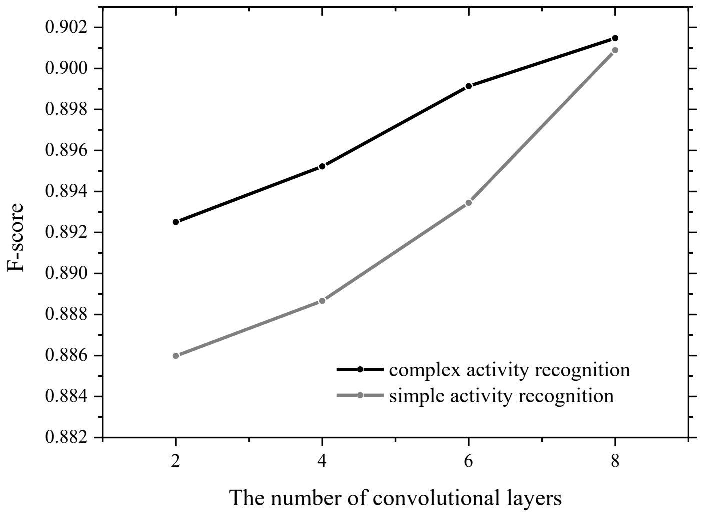

Fig. 4. The performance of complex and simple activity recognitions under different numbers of convolutional layers.

4.4 Convergence Processes

Fig. 5 illustrates the convergence processes of AROMA on Ubicomp 08 dataset. It can be seen that the classification losses on training and test datasets are extremely close to each other. They decrease along with the increase of λ and become stable gradually, which shows the stability and effectiveness of AROMA.

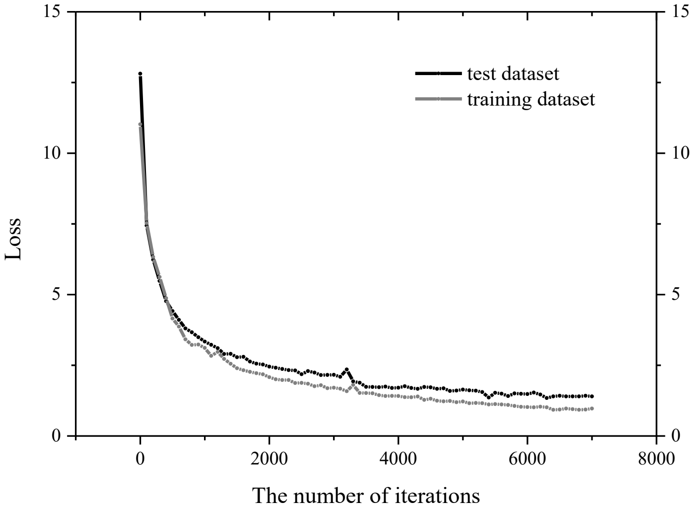

Fig. 5. The change of losses on the training and test datasets with the increase of λ.

4.5 The Experimental Results of AROMA

To evaluate the effectiveness of AROMA, we apply it on Ubicomp 08 and Opportunity datasets with parameters described earlier. Fig. 6. shows the confusion matrices of AROMA on classifying complex activities. The value in the ith row and the jth column is the proportion of complex activity samples with the ith category that are classified as the jth category. For Ubicomp 08 dataset, AROMA has better performance on classifying "commuting" and "office work" compared to "dinner activities" and "lunch routine". Both "dinner activities" and "lunch routine" tend to be misclassified as "office work". A possible reason for this behavior is that all of the three complex activities have a basic simple activity "sitting". In addition, AROMA also tends to confuse "dinner activities" with "lunch routine". It is obvious that these two complex activities usually contain many same simple activities, e.g., "fanning barbecue" and "sitting/ having a coffee", which make them difficult to be distinguished.

For Opportunity dataset, AROMA performs worse on "cleanup" and "coffee time" compared to the other three activities. Activity "cleanup" tends to be misclassified as "sandwich time", because they are based on a series of kitchen activities. Activities "coffee time" also tends to be misclassified as "sandwich time", as both of them are based on a simple activity "sitting". In addition, some samples of "coffee time" are recognized as "early morning", a possible reason for this behavior is that these two activities are both related to preparing food, including components like pouring water.

	1	2	3	4
1	70.5%	0	6.8%	22.7%
2	0	93.8%	3.1%	3.1%
3	2.5%		76.0%	21.5%
4	0.6%	0	1.1%	98.3%

(a) Ubicomp 08 dataset. 1=dinner activities, 2=commuting, 3=lunch routine, 4=office work.

	1	2	3	4	5
1	91.4%	0	6.9%	1.7%	0
2	0	78.7%	10.2%	1.5%	9.6%
3	0.3%	0.9%	95.2%	1.8%	1.8%
4	0	1.0%	7.9%	71.7%	19.4%
5	0	3.4%	2.6%	3.9%	90.1%

(b) Opportunity dataset. 1=relaxing, 2=coffee time, 3=early morning, 4=cleanup, 5=sandwich time.

Fig. 6. The confusion matrices on classifying complex activities.

4.6 Comparison with Other Methods

To demonstrate the competitive performance of AROMA, we compare it with the following methods. For fair comparison, all methods set the sample lengths of simple and complex activities the same as AROMA.

STL. STL is the single task learning version of AROMA, which optimizes the loss functions of two tasks separately. STL utilizes the same network and parameters with AROMA, but there is no shared structure.

LSTM. LSTM employs a three-layer LSTM network for both simple and complex activity recognitions. Normalized activity data are input to the LSTM network directly, and then classified by a softmax.

CNN. CNN [33] recognizes simple and complex activities using a single CNN, which contains four convolutional layers, a pooling layer, and a fully connected layer.

DeepConvLSTM. DeepConvLSTM [24] recognizes activities based on convolution operations and LSTM units. There are four convolutional layers and two layers of LSTM units, but no pooling layers. Parameters of LSTM, CNN, and DeepConvLSTM are optimized and given in Table 3.

TM. TM stands for topic model. Referring to [14], TM treats a complex activity sample as a "document", which is composed of a corpus of "words" (i.e., simple activity samples). TM firstly recognizes simple activities using a Naive Bayes based classifier, which is built on two statistical features (i.e., mean and variance) of raw data. In this way, TM can obtain lc/ls words for each document, and form a document-word matrix. Then TM applies a LDA topic model on this document-word matrix to learn a document-topic matrix, which shows the topic distributions of all documents. These topic distributions are the underlying features of complex activities. According to [14], the number of topics is set to 10.

Table 3. The parameters used in LSTM, CNN, and DeepConvLSTM for each layer. The numbers before @ refer to the parameters of convolutional kernels or pooling operations. The numbers after @ mean the number of convolutional kernels. The parameters of LSTM and DeepConvLSTM are the length of gate vectors in LSTM units.

Parameters in	Convolutional layer	Pooling layer	LSTM units
LSTM	--	--	128
CNN	5*1@ 64	2*1@64	--
DeepConvLSTM	5*1@ 64	--	128

Table 4. The F-score of all compared methods on simple activity recognition (mean±std). * indicates that our method is statistically superior to the compared method (pairwise *t*-test at 95% significance level).

	AROMA	**STL**	**LSTM**	**CNN**	**DeepConvLSTM**	**TM**
Ubicomp 08	**0.901±0.044**	0.899±0.046*	0.842±0.059*	0.883±0.051*	0.875±0.056*	0.846±0.062*
Opportunity	**0.927±0.004**	0.892±0.128*	0.865±0.021*	0.899±0.005*	0.926±0.001*	0.770±0.063*

Table 5. The F-score of all compared methods on complex activity recognition (mean±std). * indicates that our method is statistically superior to the compared method (pairwise *t*-test at 95% significance level).

	AROMA	**STL**	**LSTM**	**CNN**	**DeepConvLSTM**	**TM**
Ubicomp 08	**0.901±0.047**	0.880±0.035*	0.848±0.032*	0.827±0.087*	0.871±0.042*	0.823±0.043*
Opportunity	**0.838±0.016**	0.800±0.094*	0.731±0.016*	0.711±0.025*	0.791±0.002*	0.700±0.032*

To statistically measure the significance of performance differences, pairwise t-tests at 95% significance level are conducted between AROMA and compared methods. The classification performance of these methods is given in Tables 4 and 5, and following tendencies could be discerned:

1) On two datasets, the F-score of AROMA is higher than that of STL. It demonstrates that by exploiting the commonalities and differences between two tasks, multi-task learning can improve a model's generalization ability and performance for both simple and complex activity recognitions. In addition, the comparison between AROMA and other methods (i.e., LSTM, CNN, DeepConvLSTM, and TM, which are also single task learning based methods) can also show the superiority of multi-task learning.

2) Comparing LSTM and CNN, the former is better on complex activity recognition, and the latter is better on simple activity recognition. Simple activities are generally characterized by repeated actions or single body posture, which have limited temporal dependencies, and CNNs are proper to process. Complex activities are more complicated and have temporal dependencies, and LSTM networks are an attractive choice.

3) DeepConvLSTM is one of the latest deep learning models for HAR. The performance of STL is superior over that of DeepConvLSTM. It not only shows AROMA has a better architecture than DeepConvLSTM, but also justifies the benefits of deep residual networks.

4) TM gets worse performance compared to deep learning methods. It indicates that manually designed features are poorer at capturing internal data properties compared to deep features. Deep learning methods can discover deep features automatically, which can own more discriminative power.

By using a CNN, a LSTM network, and a shared structure, AROMA can recognize simple and complex activities jointly, but it has a high demand on datasets, i.e., having two-level activity labels. In addition, our work also has some limitations. First, the design space of shared structure is not fully explored. Second, multimodal data are not processed specifically. Third, the performance of AROMA is not systematically investigated.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we propose AROMA, a deep multi-task learning method to solve simple and complex activity recognition tasks jointly. For simple activity recognition task, AROMA uses a CNN to discover deep features automatically, which are discriminative and task dependent. For complex activity recognition task, the deep features after the CNN are input into a LSTM network. In this way, AROMA can make full use of the temporal context of activity data. The two tasks have a shared structure (i.e., the CNN and the deep feature layer), which can benefit both tasks and improve the generalization ability of AROMA. We evaluate AROMA on two public datasets, and the experimental results show that AROMA is able to yield a competitive performance in both simple and complex activity recognitions.

In the future, we will extend our work in the following directions. First, the shared structure of two tasks can be set in any hidden layer of the network, and we will improve our model by applying a more intelligent network (e.g., cross-stitch network), which can learn the shared structure automatically. Second, we will employ specific sub-structures for different modality data. Third, we will study the performance of AROMA with more datasets and investigate the effects of other factors (e.g., the position and number of sensors).

ACKNOWLEDGMENTS

This work is supported by China Knowledge Centre for Engineering Sciences and Technology under Grant No. CKCEST-2014-1-5, the National Natural Science Foundation of China under Grant No. 61332017, and the Science and Technology Department of Zhejiang Province under Grant No. 2015C33002.

REFERENCES

[1] Vincent Van Asch. 2013. Macro- and micro-averaged evaluation measures. *Belgium: University of Antwerp, Technical Report*, (2013).

[2] Ling Bao and Stephen S. Intille. 2004. Activity recognition from user-annotated acceleration data. In *Proceedings of the second International Conference on Pervasive Computing (Pervasive'04).* Springer, 1-17.

[3] Ulf Blanke and Bernt Schiele. 2009. Daily routine recognition through activity spotting. *Location and Context Awareness* (2009), 192-206.

[4] Ulf Blanke and Bernt Schiele. 2010. Remember and transfer what you have learned-recognizing composite activities based on activity spotting. In *Proceedings of the fourteenth International Symposium on Wearable Computers (ISWC'10).* IEEE, 1-8.

[5] Saisakul Chernbumroong, Shuang Cang, Anthony Atkins, and Hongnian Yu. 2013. Elderly activities recognition and classification for applications in assisted living. *Expert Systems with Applications* 40, 5 (2013), 1662-1674.

[6] Stefan Dernbach, Barnan Das, Narayanan C. Krishnan, Brian L. Thomas, and Diane J. Cook. 2012. Simple and complex activity recognition through smart phones. In *Proceedings of the eighth International Conference on Intelligent Environments (IE'12).* IEEE, 214-221.

[7] Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jurgen Schmidhuber. 2017. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems* 28, 10 (2017), 2222-2232.

[8] Yu Guan and Thomas Plötz. 2017. Ensembles of deep LSTM learners for activity recognition using wearables. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT'17).* ACM, 11-28.

[9] Piyush Gupta and Tim Dallas. 2014. Feature selection and activity recognition system using a single tri-axial accelerometer. *IEEE Transactions on Biomedical Engineering* 61, 6 (2014), 1780-1786.

[10] Nils Y. Hammerla, Shane Halloran, and Thomas Plötz. 2016. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the twenty-fifth International Joint Conference on Artificial Intelligence (IJCAI'16).* Morgan Kaufmann,1533-1540.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the thirty-fourth IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16).* IEEE, 770-778.

[12] Zhenyu He and Lianwen Jin. 2008. Activity recognition from acceleration data using AR model representation and SVM. In *Proceedings of the first International Conference on Machine Learning and Cybernetics (ICMLC'08)*. IEEE, 2245-2250.

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735-1780.

[14] Tâm Huynh, Mario Fritz, and Bernt Schiele. 2008. Discovery of activity patterns using topic models. In *Proceedings of the tenth International Conference on Ubiquitous Computing (Ubicomp'08)*. ACM, 10-19.

[15] Abhishek Kumar and Hal Daume. 2012. Learning task grouping and overlap in multi-task learning. In *Proceedings of the twenty-ninth International Conference on Machine Learning (ICML'12)*. ACM, 1383-1390.

[16] Óscar D. Lara, Alfredo J. Pérez, Miguel A. Labrador, and José D. Posada. 2012. Centinela: A human activity recognition system based on acceleration and vital sign data. *Pervasive and Mobile Computing* 8, 5 (2012), 717-729.

[17] Yann LeCun and Yoshua Bengio. 1995. Convolutional networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks* 10, (1995), 3361.

[18] Li Liu, Yuxin Peng, Ming Liu, and Zigang Huang. 2015. Sensor-based human activity recognition system with a multilayered model using time series shapelets. *Knowledge-Based Systems* 90, (2015), 138-152.

[19] Li Liu, Yuxin Peng, Shu Wang, Ming Liu, and Zigang Huang. 2016. Complex activity recognition using time series pattern dictionary learned from ubiquitous sensors. *Information Sciences* 340, (2016), 41-57.

[20] Mingqi Lv, Ling Chen, Tieming Chen, and Gencai Chen. 2018. Bi-view semi-supervised learning based semantic human activity recognition using accelerometers. *IEEE Transactions on Mobile Computing* (2018).

[21] Takuya Maekawa, Daisuke Nakai, Kazuya Ohara, and Yasuo Namioka. 2016. Toward practical factory activity recognition: Unsupervised understanding of repetitive assembly work in a factory. In *Proceedings of the eighteenth International Conference on Ubiquitous Computing (Ubicomp'16)*. ACM, 1088-1099.

[22] Opportunity dataset. Available online: http://www.opportunity-project.eu/node/56

[23] Francisco Javier Ordóñez Morales and Daniel Roggen. 2016. Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations. In *Proceedings of the twentieth International Symposium on Wearable Computers (ISWC'16)*. IEEE, 92-99.

[24] Francisco Javier Ordóñez and Daniel Roggen. 2016. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16, 1 (2016), 115.

[25] Liangying Peng, Ling Chen, Xiaojie Wu, Haodong Guo, and Gencai Chen. 2017. Hierarchical complex activity representation and recognition using topic model and classifier level fusion. *IEEE Transactions on Biomedical Engineering* 64, 6 (2017), 1369-1379.

[26] Daniel Roggen, Alberto Calatroni, Mirco Rossi, et al. 2010. Collecting complex activity data sets in highly rich networked sensor environments. In *Proceedings of the seventh IEEE International Conference on Networked Sensing Systems (INSS'10)*. IEEE, 233–240.

[27] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv*:1706.05098, (2017).

[28] Hava T. Siegelmann. 1995. Recurrent neural networks. *Computer Science Today* (1995), 29-45.

[29] Xu Sun, Hisashi Kashima, Ryota Tomioka, Naonori Ueda, and Ping Li. 2011. A new multi-task learning method for personalized activity recognition. In *Proceedings of the eleventh International Conference on Data Mining (ICDM'11)*. IEEE, 1218-1223.

[30] Xu Sun, Hisashi Kashima, and Naonori Ueda. 2013. Large-scale personalized human activity recognition using online multitask learning. *IEEE Transactions on Knowledge and Data Engineering* 25, 11 (2013), 2551-2563.

[31] Ubicomp 08 Dataset. Available online: https://www.mpi-inf.mpg.de/departments/computer-vision-and-multimodal-computing/software-and-datasets/

[32] Rui Wang, Fanglin Chen, Zhenyu Chen, Tianxing Li, Gabriella Harari, Stefanie Tignor, Xia Zhou, Dror Ben-Zeev, and Andrew T. Campbell. 2014. StudentLife: Assessing mental health, academic performance and behavioral trends of college students using smartphones. In *Proceedings of the sixteenth International Conference on Ubiquitous Computing (Ubicomp'14)*. ACM, 3-14.

[33] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. 2015. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the twenty-fourth International Joint Conference on Artificial Intelligence (IJCAI'15)*. Morgan Kaufmann, 3995-4001.

[34] Ming Zeng, Le T. Nguyen, Bo Yu, Ole J. Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. 2014. Convolutional neural networks for human activity recognition using mobile sensors. In *Proceedings of the sixth International Conference on Mobile Computing, Applications and Services (MobiCASE'14)*. IEEE, 197-205.

[35] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. 2014. Facial landmark detection by deep multi-task learning. In *Proceedings of the twelfth European Conference on Computer Vision (ECCV'14)*. Springer, 94-108.

Received November 2017; revised February 2018; accepted April 2018