METIER: A Deep Multi-Task Learning Based Activity and User Recognition Model Using Wearable Sensors

LING CHEN*, Zhejiang University
YI ZHANG, Zhejiang University
LIANGYING PENG, Zhejiang University

Activity recognition (AR) and user recognition (UR) using wearable sensors are two key tasks in ubiquitous and mobile computing. Currently, they still face some challenging problems. For one thing, due to the variations in how users perform activities, the performance of a well-trained AR model typically drops on new users. For another, existing UR models are powerless to activity changes, as there are significant differences between the sensor data in different activity scenarios. To address these problems, we propose METIER (deep multi-task learning based activity and user recognition) model, which solves AR and UR tasks jointly and transfers knowledge across them. User-related knowledge from UR task helps AR task to model user characteristics, and activity-related knowledge from AR task guides UR task to handle activity changes. METIER softly shares parameters between AR and UR networks, and optimizes these two networks jointly. The commonalities and differences across tasks are exploited to promote AR and UR tasks simultaneously. Furthermore, mutual attention mechanism is introduced to enable AR and UR tasks to exploit their knowledge to highlight important features for each other. Experiments are conducted on three public datasets, and the results show that our model can achieve competitive performance on both tasks.

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

Additional Key Words and Phrases: Activity recognition, user recognition, multi-task learning, mutual attention mechanism

ACM Reference Format:

Ling Chen, Yi Zhang, and Liangying Peng. 2020. METIER: A Deep Multi-Task Learning Based Activity and User Recognition Model Using Wearable Sensors. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 1, Article 5 (March 2020), 18 pages. https://doi.org/10.1145/3381012

*This is the corresponding author

Authors' addresses: 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; Y. Zhang, College of Computer Science and Technology, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China, zhangyi1995@zju.edu.cn; L. Peng, College of Computer Science and Technology, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China, lyoare@zju.edu.cn.

1 INTRODUCTION

Activity recognition (AR) and user recognition (UR) using wearable sensors are two key tasks in ubiquitous and mobile computing, which can provide personalized support for a variety of applications, e.g., health support [15], skill assessment [18], and biometric authentication [8].

AR using wearable sensors aims to recognize users' ongoing activities by analyzing the data from on-body sensors. Most AR methods merge all the data from available training users, and exploit activity labels as supervisory signals to train a model for new users [2, 12, 20, 27]. The variations in how users perform activities are usually ignored. Existing studies [5, 16] have shown that different users perform activities in different ways, which means there is a distribution shift among the data from different users. Significant performance degradation typically occurs when a well-trained model is applied to new users. The efforts on this problem can be roughly classified into two categories. The first category picked out user-independent features for modeling [17, 36]. These methods can improve model generalization, but also weaken the discriminative power of the features. The second category built a personalized model for each user [14, 19, 21, 41]. These methods can achieve good performance, but the high cost of collecting data and training different models for different users limits their popularity.

UR using wearable sensors is intended to identify the current user by analyzing the data from on-body sensors. Most UR methods [4, 8, 39] can identify users by only walking data. Although satisfactory results have been reported, limited activity scenarios still restrict their application in daily life. Extending the activity scenarios they support from walking to other activities is valuable but also challenging, as there are significant differences between the sensor data in different activity scenarios.

To address these problems mentioned above, we propose METIER (deep multi-task learning based activity and user recognition) model, which solves AR and UR tasks jointly and transfers knowledge across them. For AR task, by introducing user-related knowledge from UR task, it can take users' characteristics into account while recognizing activities. For UR task, different features need to be used in different activity scenarios. With the guidance of activity-related knowledge from AR task, UR task can automatically adapt to current activity. The main contributions of this paper are summarized as follows:

(1) Propose METIER model to solve AR and UR tasks jointly, which softly shares parameters between AR and UR networks, and optimizes these two networks together. The commonalities and differences across tasks are exploited to promote both tasks simultaneously.

(2) Introduce mutual attention mechanism between AR and UR networks, which enables AR and UR tasks to exploit their knowledge to highlight important features for each other. The highlighted features change automatically based on current activity or user.

(3) Conduct experiments on three public datasets, SBHAR [33], UniMiB [22], and REALDISP [1], and the results show that the proposed model has competitive performance on both tasks.

The remainder of this paper is organized as follows: Section 2 presents the related work. Section 3 describes the proposed model in detail. The experiments are given in Section 4. Finally, Section 5 concludes this paper.

2 RELATED WORK

2.1 Activity Recognition Using Wearable Sensors

AR using wearable sensors can be classified into two categories, i.e., non-personalized AR and personalized AR. Non-personalized AR trains a general model for all users. The common practice is to merge all the data from available training users and use activity labels as supervisory signals to train a general model [2, 11, 20, 29]. Due to the variations in how users perform activities, it faces the problem of model generalization. Some methods [17, 36, 38, 44] improve non-personalized AR by picking out user-independent features for modeling. For instance, Saputri et al. [36] proposed a three-stage genetic algorithm-based feature selection method to select user-independent features. These improved methods increase model applicability on new users, but also

weaken the discriminative power of the features, especially when there are large differences in how users perform activities.

Personalized AR trains a private model for each user [19, 34, 40, 50]. Since it is difficult to collect enough training data for all users, personalized AR typically uses some machine learning (ML) techniques, e.g., multi-task learning [40, 41], transfer learning [21, 32, 34, 50], and domain adaptation [19], to reduce the need for training data. For instance, Sun et al. [40] proposed an online multi-task learning based personalized AR method, in which each task corresponds to a person. The similarities among different tasks are mined and exploited to improve learning efficiency and reduce the requirement of training samples. Rokni et al. [34] developed a convolutional neural network (CNN) based transfer learning model for personalized AR, which reuses the lower layers of a well-trained network and retrains its upper layers with a small amount of labeled data from a new user. Reiss and Stricker [32] constructed a personalized AR model consisting of a set of classifiers, which retrains only the weights of these classifiers with a small amount of labeled data from a new user. These methods reduce the need for training data to a certain extent, but their expensive training cost is still inevitable.

In this work, non-personalized AR model is jointly learned with UR model. For one hand, it avoids the high cost of personalized AR. For another, by introducing user-related knowledge from UR task, AR task can take users' characteristics into account while recognizing activities.

2.2 User Recognition Using Wearable Sensors

Existing UR methods using wearable sensors can be roughly classified into two categories, i.e., template based methods and ML based methods. The former builds templates for each user, and identifies users by cross-comparing templates with input samples [7, 8, 35]. For instance, Derawi et al. [8] proposed an acceleration sensor based UR method, which creates an average gait cycle template for each user and identifies target user by comparing the dynamic time warping (DTW) distance between the templates and input samples. Del Pozo et al. [7] proposed a speed-independent UR method, which extracts average templates from the gait instances at different velocities and exploits Euclidean and DTW distance to identify users. The latter learns a mapping from predesigned features to user labels [4, 25, 39]. For instance, Sprager and Zazula [39] proposed a cumulant-based method, which calculates the cumulants for gait cycles as features, and trains a support vector machine classifier to identify users. Choi et al. [4] designed six features from the change rate of acceleration data, and used a k nearest neighbors (kNN) classifier to identify users.

Most of prior methods identify users by walking data, which is limited in daily life. Currently, a small amount of work attempts to extend the supported activity scenarios of UR from walking to other activities [9, 48]. For instance, Yao et al. [48] proposed a deep learning (DL) based framework to identify users, where DL helps to automatically extract distinct features that fit well to not only walking but also other activities (e.g., biking and climbing stairs). Elkader et al. [9] presented a ML based UR method, which identifies users in twenty representative activity scenarios. These methods face the problem of significant differences in sensor data caused by activity changes.

To deal with activity changes, we train UR and AR models jointly. Activity-related knowledge from AR task can be transferred to UR task and help it automatically adapt to current activity.

2.3 Multi-Task Learning

Multi-task learning is a subfield of ML that is contrary to single-task learning, in which multiple tasks are solved jointly. The commonalities and differences among these tasks are mined and used to improve performance and generalization. There are two common practices to perform multi-task learning: soft parameter sharing [26, 47] and hard parameter sharing [3, 49]. The former keeps each task with its own model and parameters, and places constrains on the parameters of multiple tasks. For instance, Obozinski et al. [26] represented the parameters of multiple tasks as a matrix, and minimized its $l_{2,1}$ norm to encourage multiple tasks to have similar parameter sparsity patterns. The latter shares some parameters among all tasks. For

instance, Caruana [3] proposed a classical deep multi-task learning framework, which defines deep neural network (DNN) with shared lower layers.

There is some work that uses multi-task learning in AR and UR studies. For instance, Peng et al. [30] presented a deep multi-task learning method that solves simple and complex AR tasks jointly. Papavasileiou et al. [28] proposed a multi-task learning based UR model, where each task corresponds to a binary classifier. These efforts demonstrate the effectiveness of multi-task learning in AR and UR studies. Iwasawa et al. [17] attempted to perform a regular AR task and an adversarial UR task jointly. AR and UR networks share lower layers. Adversarial UR task aims to learn user-independent features to make UR difficult, and regular AR task recognizes activities based on these user-independent features. However, adversarial UR task also weakens the discriminative power of the features.

In this work, we propose a deep multi-task learning based activity and user recognition model, which introduces soft parameter sharing and mutual attention mechanism to further utilize the correlation between AR and UR tasks.

3 MODEL

3.1 Problem Definitions

For AR and UR tasks, it is assumed that one or more wearable sensors are attached on human body to collect data. Raw streaming data are preprocessed and segmented to produce equal-sized time windows, which consist of multiple time series returned by all wearable sensor channels during a fixed time interval. Let W, A, and U denote time window set, activity set, and user set, respectively. A dataset DS can be formulated as:

$$DS = \{(w, a, u)\}, w \in W, a \in A, u \in U \tag{1}$$

Definition 1 (AR task): Given a dataset DS, AR task is to learn a mapping function $h: W \rightarrow A$ from the samples in DS. For any unseen sample (w, a, u), $h(w)$ and a should be as similar as possible.

Definition 2 (UR task): Given a dataset DS, UR task is to learn a mapping function $g: W \rightarrow U$ from the samples in DS. For any unseen sample (w, a, u), $g(w)$ and u should be as similar as possible.

3.2 Framework

As shown in Fig. 1, METIER contains two networks, i.e., AR and UR networks, which perform AR and UR tasks, respectively, and there are some interactions between them. In each network, a time window is first transformed through a CNN consisting of three convolutional layers and one max pooling layer, and then a one-layer bidirectional long short-term memory (bi-LSTM) follows and generates task-specific features. Finally, attention mechanism highlights important features by weights, and these weighted features are used for classification. Particularly, we softly share parameters between the corresponding convolutional layers of AR and UR networks, and the commonalities and differences across tasks are exploited to promote these two tasks simultaneously. Furthermore, we introduce mutual attention mechanism [42] between these two networks instead of traditional attention mechanism, which enables AR and UR tasks to exploit their knowledge to highlight important features for each other. The details of CNN, bi-LSTM, and mutual attention mechanism are introduced in Sections 3.3-3.5.

3.3 CNN

Traditional AR and UR models usually work on hand-crafted features, e.g., mean and standard deviation, which require expert knowledge and typically result in information loss. We exploit CNN to extract deep features from sensor data, where manual feature engineering is replaced by automated feature learning. The stacked convolutional layers enhance features layer by layer, generating discriminative and task dependent deep features.

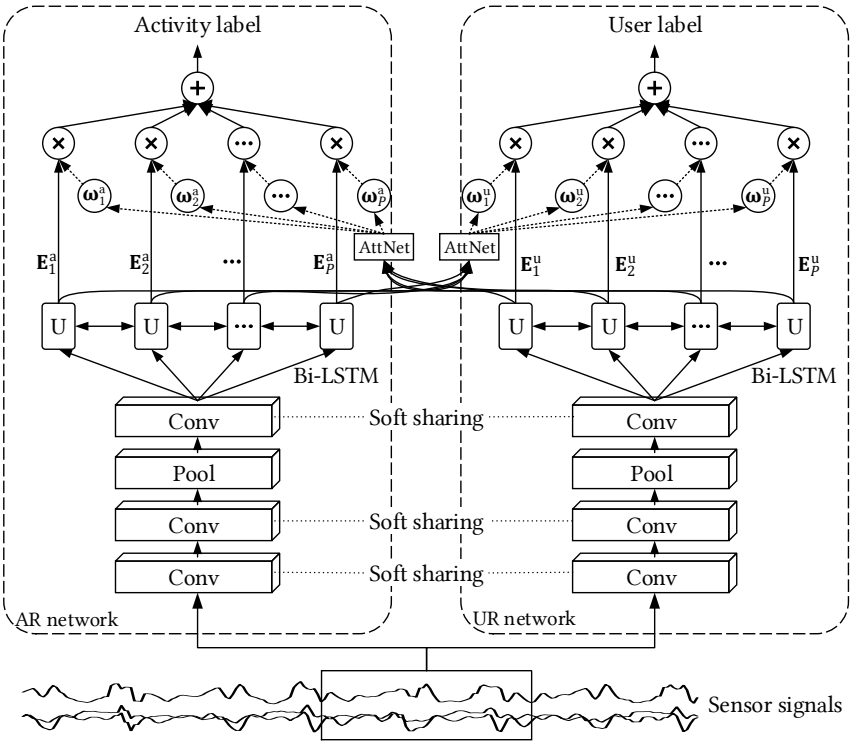

Fig. 1. The overall framework of METIER. "Conv" and "Pool" refer to convolutional layer and max pooling layer, respectively. "U" denotes the unit of bi-LSTM, and "AttNet" is attention network, which outputs weight $\boldsymbol{\omega}$ to differentiate the importance of features $\mathbf{E}$. "$\oplus$" and "$\otimes$" denote pointwise addition and multiplication, respectively.

CNN is a class of deep feed-forward artificial neural networks. It is commonly used as a feature extractor in time-series data processing [27, 46]. In general, CNN is a multi-layer network that alternately stacks convolutional layers and pooling layers. Convolution filters are the core of a convolutional layer, which convolve with inputs to produce feature maps. Typically, each convolutional layer generates multiple feature maps. By stacking multiple convolutional layers, deep features will be extracted layer by layer. Convolution operation is given formally by Equation 2:

$$\mathbf{X}_j^{l+1} = \sigma\left(\mathbf{b}_j^l + \sum_{i=1}^{T^l}\left(\mathbf{K}_{i,j}^l * \mathbf{X}_i^l\right)\right) \tag{2}$$

where $\mathbf{X}_i^l$ denotes the i-th feature map in layer l, and $\mathbf{K}_{i,j}^l$ is the convolution filter convolved over feature map $\mathbf{X}_i^l$ to produce the j-th feature map in layer $l+1$. "$*$" denotes convolution operation. T^l represents the number of feature maps in layer l. $\mathbf{b}_j^l$ is a bias, and σ is a non-linear activation function. Pooling layer is an important part of CNN, where non-linear down-sampling is performed. It partitions input into a set of non-overlapping regions, and outputs a value for each region. Pooling layer is used to reduce the size and computation of a network, and control overfitting. Max pooling and average pooling are two commonly used pooling operations, which are given formally by Equations 3 and 4, respectively:

$$\mathbf{X}_{i,j}^{l+1} = max_{k=0}^{s-1}\left(\mathbf{X}_{i,j\times s+k}^l\right) \tag{3}$$

$$\mathbf{X}_{i,j}^{l+1} = \frac{1}{s}\sum_{k=0}^{s-1}\left(\mathbf{X}_{i,j\times s+k}^l\right) \tag{4}$$

where $\mathbf{X}_{i,j}^l$ denotes the j-th value in feature map $\mathbf{X}_i^l$, and s denotes the size of pooling region.

The design of CNN is inspired by related work [13, 27, 46], and we use a CNN consisting of three convolutional layers and one max pooling layer to extract deep features from sensor data. The number of convolutional layers is a key parameter. Although more convolutional layers can theoretically extract more discriminative features, but also increase the difficulty of training. We use three convolutional layers for feature extraction, which can achieve a balance. To obtain complete features, each convolutional layer generates 32 feature maps. We add one max pooling layer between the convolutional layers to reduce network size and control overfitting. The sizes of convolution filters and pooling regions follow the most common settings in related work [46], and are set to 1×5 and 1×2, respectively. The convolution and max pooling operations are performed along the time axis, which keeps each sensor channel independent in CNN and avoids the data compatibility issues caused by the fusion of sensor data with different properties. According to the finding in [24], i.e., the CNN sharing convolution filters on all channels has a low dependency on the amount of training data, we share convolution filters on all channels. Rectified linear unit serves as the non-linear activation function in each convolutional layer, which is given formally by Equation 5:

$$\sigma(x) = max(x, 0) \tag{5}$$

The structure of CNN is shown in Fig. 2. A time window is first transformed through two convolutional layers. Then, the outputs of previous step are put through a max pooling layer. Following one convolutional layer, the outputs of the CNN are generated.

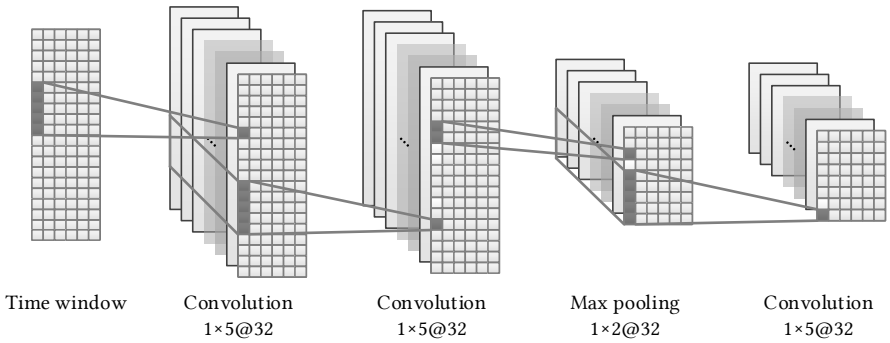

Fig. 2. The structure of CNN. The numbers before @ refer to the size of convolution filters or pooling regions, and the numbers after @ mean the number of feature maps.

We softly share parameters between the corresponding convolutional layers of AR and UR networks using a tensor decomposition based strategy [47], which can learn what and how much to share. In a convolutional layer, the parameters that need to be kept and updated are essentially convolution filters. Tensor decomposition based parameter sharing strategy considers convolution filters to be generated by latent parameters, some of which are shared by AR and UR networks. Therefore, in our model, convolutional layers keep the latent parameters rather than the original convolution filters. In the forward pass, convolution filters are generated from these latent parameters first, and then used for inference. In the backward pass, instead of convolution filters, these latent parameters are updated. When training one network, the updates of shared latent parameters will affect the generated convolution filters of another network as well, so that knowledge can be transferred across tasks. Specifically, in each convolutional layer of METIER, the stack of the convolution filters in AR and UR networks can be represented by a 4-way tensor $\mathbf{W}$ with the size of $d_1 \times d_2 \times d_3 \times d_4$ ($d_4 = 2$), which is generated by the reverse operation of Tucker decomposition, as shown in Equation 6:

$$\mathbf{W} = \mathbf{S} \cdot_{(1,2)} \mathbf{R}^{(1)} \cdot_{(1,2)} \mathbf{R}^{(2)} \cdot_{(1,2)} \mathbf{R}^{(3)} \cdot_{(1,2)} \mathbf{R}^{(4)} \tag{6}$$

where $\mathbf{S}$ is a core tensor with the size of $v_1 \times v_2 \times v_3 \times v_4$, and $\mathbf{R}^{(i)}$ denotes a matrix with the size of $d_i \times v_i$. The subscript of operator "$\cdot_{(i,j)}$" denotes the axes where dot product is performed. In this case, $\mathbf{R}^{(4)}$ represents task-specific latent parameters, and the rest are shared latent parameters. The ranks and initial values of these latent parameters are determined as follows: First train AR and UR networks without soft parameter sharing. Then stack the convolution filters of the corresponding convolutional layers of AR and UR networks into 4-way tensors. Decompose these tensors using Tucker decomposition to produce initial latent parameters.

3.4 Bi-LSTM

Temporal dependency discovery is the key to time-series data processing. Existing DL-based work attempts to introduce RNN to learn the temporal dependencies from past to future. The temporal dependencies from future to past are usually ignored. In METIER, we use bi-LSTM to learn temporal dependencies from both directions.

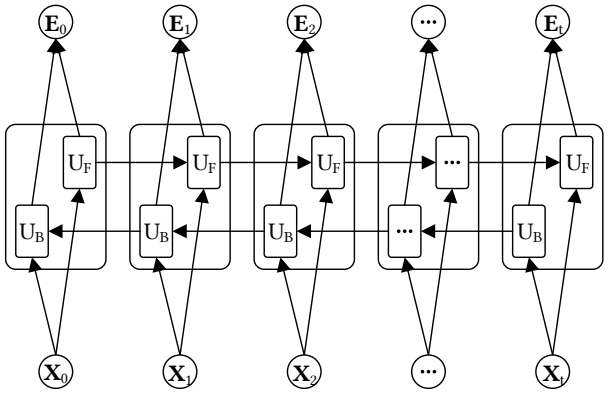

Fig. 3. The workflow of bi-LSTM. "U_F" and "U_B" refer to the forward and backward processing units.

In METIER, a one-layer bi-LSTM is used to capture the temporal dependencies in sensor data. Fig. 3 shows the workflow of bi-LSTM, which takes the outputs of CNN as input, and outputs task-specific features. The outputs of CNN are first flattened into a vector sequence $\mathbf{X}$, where the time axis remains and the other axes are flattened, and then input into the bi-LSTM. Two units process the input sequence step by step from two directions. In each step, in addition to the current input, the state of the previous step is also entered into the processing unit. The state is passed step by step during processing and encodes the temporal dependencies in input sequence. Finally, the outputs of both processing units are concatenated into a vector sequence $\mathbf{E}$ as task-specific features.

3.5 Mutual Attention

AR and UR models usually face environment changes (user or activity scenario changes). Due to the significant differences between different environments, it is hard to find a fixed set of features that adapt to all environments. Attention mechanism [23] seems to be a promising solution to this problem, which enables a neural network to automatically focus on important features by adjusting importance weights. Attention mechanism has achieved great success. For example, self-attention mechanism [45] has shown its superiority in many tasks. However, traditional attention mechanisms are typically designed for a single task, and the correlation between multiple tasks cannot be utilized. Therefore, in METIER, we introduce mutual attention mechanism [42] between AR and UR networks, which can exploit the knowledge from one task to highlight the important features of another task. It consists of two parts that cooperate with each other, and these two

parts are responsible for highlighting important features for AR and UR, respectively. Different from traditional attention mechanism, the feature highlight for one task is guided by the knowledge of another task.

AR task is always troubled by the variations in how users perform activities, and the performance of a well-trained AR model typically drops on new users. In order to improve the generalization, user features learned by UR task are introduced into AR network, which encode information related to user characteristics. Mutual attention mechanism learns a mapping from user features to the weights of activity features, which highlights the features that are suit to the current user. It can be described formally by Equations 7-9:

$$\boldsymbol{\rho}^{\mathrm{a}} = \mu^{\mathrm{a}}(\mathbf{E}^{\mathrm{u}}) \tag{7}$$

$$\boldsymbol{\omega}_i^{\mathrm{a}} = \frac{\boldsymbol{\rho}_i^{\mathrm{a}}}{\sum_{i=1}^{P} exp(\boldsymbol{\rho}_i^{\mathrm{a}})} \tag{8}$$

$$\mathbf{c}^{\mathrm{a}} = \sum_{i=1}^{P} (\boldsymbol{\omega}_i^{\mathrm{a}} \times \mathbf{E}_i^{\mathrm{a}}) \tag{9}$$

where $\mathbf{E}^{\mathrm{u}}$ denotes user features, and μ^{a} maps $\mathbf{E}^{\mathrm{u}}$ to the weight vector $\boldsymbol{\rho}^{\mathrm{a}}$ that differentiates the importance of activity features. Activity features are represented as a vector sequence of length P, each step of which $\mathbf{E}_i^{\mathrm{a}}$ corresponds to the i-th dimension of weight vector $\boldsymbol{\rho}_i^{\mathrm{a}}$. $\boldsymbol{\omega}^{\mathrm{a}}$ is the normalized weight vector and $\mathbf{c}^{\mathrm{a}}$ is the weighted sum of activity features, which is used for activity classification. Specifically, μ^{a} is implemented with a two-layer fully connected neural network. The first layer contains 256 cells, and the number of cells in the second layer is determined by the length of activity features.

UR task is expected to adapt to a variety of activity scenarios, which faces the problem of significant differences between the sensor data in different activity scenarios. To address this problem, we exploit mutual attention mechanism to highlight different user features for identification in different activity scenarios. Activity features learned by AR task are introduced, which encode information related to activity scenarios. Mutual attention mechanism learns a mapping from activity features to the weights of user features, and adjusts these weights automatically according to the current activity scenario. It can be described formally by Equations 10-12:

$$\boldsymbol{\rho}^{\mathrm{u}} = \mu^{\mathrm{u}}(\mathbf{E}^{\mathrm{a}}) \tag{10}$$

$$\boldsymbol{\omega}_i^{\mathrm{u}} = \frac{\boldsymbol{\rho}_i^{\mathrm{u}}}{\sum_{i=1}^{P} exp(\boldsymbol{\rho}_i^{\mathrm{u}})} \tag{11}$$

$$\mathbf{c}^{\mathrm{u}} = \sum_{i=1}^{P} (\boldsymbol{\omega}_i^{\mathrm{u}} \times \mathbf{E}_i^{\mathrm{u}}) \tag{12}$$

where $\mathbf{E}^{\mathrm{a}}$ denotes activity features and $\boldsymbol{\rho}^{\mathrm{u}}$ denotes the weight vector that differentiates the importance of user features. μ^{u} is the mapping from $\mathbf{E}^{\mathrm{a}}$ to $\boldsymbol{\rho}^{\mathrm{u}}$, which is implemented using the same network structure as μ^{a}. $\boldsymbol{\omega}^{\mathrm{u}}$ is the normalized weight vector. User features are represented as a vector sequence of length P, and $\mathbf{E}_i^{\mathrm{u}}$ is its i-th step. $\mathbf{c}^{\mathrm{u}}$ is the weighted sum of user features.

3.6 Training

Cross entropy is used as loss function in our model. Two losses $\mathcal{L}_{\mathrm{a}}$ and $\mathcal{L}_{\mathrm{u}}$ are defined for AR and UR networks, respectively, which are given formally by Equations 13 and 14:

$$\mathcal{L}_{\mathrm{a}} = -\frac{1}{Q}\sum_{i=1}^{Q} \mathbf{y}_i^{\mathrm{a}} \cdot log\left(f^{\mathrm{a}}(w_i)\right) \tag{13}$$

$$\mathcal{L}_{\mathrm{u}} = -\frac{1}{Q}\sum_{i=1}^{Q} \mathbf{y}_i^{\mathrm{u}} \cdot log\left(f^{\mathrm{u}}(w_i)\right) \tag{14}$$

where w_i denotes a time window. f^{a} and f^{u} represent our model, which map time window w_i to the probabilities of activity and user, respectively. $\mathbf{y}_i^{\mathrm{a}}$ and $\mathbf{y}_i^{\mathrm{u}}$ are the one-hot coding of actual activity and user labels for time window w_i. Q denotes the number of time windows in training set. The training of METIER

can be divided into two phases: latent parameter initialization phase and training phase. In latent parameter initialization phase, a simplified model of METIER without soft parameter sharing is learned and used for the initialization of METIER. Specifically, in each convolutional layer, we stack the convolution filters of AR and UR networks into a tensor, and then decompose it to initialize shared and task-specific latent parameters. In training phase, we combine two losses above into a global loss $\mathcal{L}_{\mathrm{g}}$ and use it to update METIER model. $\mathcal{L}_{\mathrm{g}}$ is given formally by Equation 15:

$$\mathcal{L}_{\mathrm{g}} = \mathcal{L}_{\mathrm{a}} + \alpha \times \mathcal{L}_{\mathrm{u}} \tag{15}$$

where α is a weight factor to balance $\mathcal{L}_{\mathrm{a}}$ and $\mathcal{L}_{\mathrm{u}}$. We represent the parameters of CNN, bi-LSTM, and mutual attention mechanism in AR network with $\theta_{\mathrm{a,CNN}}$, $\theta_{\mathrm{a,LSTM}}$, and $\theta_{\mathrm{a,MA}}$, respectively, and represent that in UR network with $\theta_{\mathrm{u,CNN}}$, $\theta_{\mathrm{u,LSTM}}$, and $\theta_{\mathrm{u,MA}}$, respectively. θ_{la} and θ_{lu} denote task-specific latent parameters, and θ_{ls} denotes shared latent parameters. The detailed training process is shown in Algorithm 1:

ALGORITHM 1: The training process of METIER

Input: Dataset: D; Learning rate: l; Numbers of iterations: λ_{i}, λ_{t}
Output: Trained METIER

#Latent parameter initialization
Initialize a simplified model of METIER without soft parameter sharing
Repeat
    Randomly select fixed number of training samples to construct mini-batch
    #Forward Propagation
    Calculate $\mathcal{L}_{\mathrm{a}}$ using Equation 13
    #Backward Propagation
    Update the parameters of AR network $\theta_{\mathrm{a}} \leftarrow \left(\theta_{\mathrm{a}} - l \times \frac{\partial \mathcal{L}_{\mathrm{a}}}{\partial \theta_{\mathrm{a}}}\right)$, $\theta_{\mathrm{a}} = \{\theta_{\mathrm{a,CNN}}, \theta_{\mathrm{a,LSTM}}, \theta_{\mathrm{a,MA}}\}$
    #Forward Propagation
    Calculate $\mathcal{L}_{\mathrm{u}}$ using Equation 14
    #Backward Propagation
    Update the parameters of UR network $\theta_{\mathrm{u}} \leftarrow \left(\theta_{\mathrm{u}} - l \times \frac{\partial \mathcal{L}_{\mathrm{u}}}{\partial \theta_{\mathrm{u}}}\right)$, $\theta_{\mathrm{u}} = \{\theta_{\mathrm{u,CNN}}, \theta_{\mathrm{u,LSTM}}, \theta_{\mathrm{u,MA}}\}$
Until Reach the number of iterations λ_{i}
Decompose the stack of $\theta_{\mathrm{a,CNN}}$ and $\theta_{\mathrm{u,CNN}}$ to initialize shared latent parameters θ_{ls} and task-specific latent parameters θ_{la} and θ_{lu}
#Training
Repeat
    Randomly select fixed number of training samples to construct mini-batch
    #Forward Propagation
    Calculate $\mathcal{L}_{\mathrm{g}}$ using Equation 15
    #Backward Propagation
    Update the parameters of METIER $\theta \leftarrow \left(\theta - l \times \frac{\partial \mathcal{L}_{\mathrm{g}}}{\partial \theta}\right)$, $\theta = \{\theta_{\mathrm{a,LSTM}}, \theta_{\mathrm{a,MA}}, \theta_{\mathrm{u,LSTM}}, \theta_{\mathrm{u,MA}}, \theta_{\mathrm{ls}}, \theta_{\mathrm{la}}, \theta_{\mathrm{lu}}\}$
Until Convergence or reach the number of training iterations λ_{t}
Return Trained METIER

4 EXPERIMENTS

4.1 Datasets

We evaluate METIER on three public datasets, SBHAR [33], UniMiB [22], and REALDISP [1], all of which provide a sufficient number of activity and user labels to support both AR and UR studies.

SBHAR dataset provides a total of around 5 hours of smartphone based data for AR and UR studies. 30 users (aged 19 to 48) participated in data collection. They were asked to wear a smartphone on the waist and perform 12 different activities following a predefined protocol that specifies the process of performing activities. The data from built-in accelerometer and gyroscope were collected and labeled with both activity and user labels. This dataset is publicly available and can be downloaded from [37].

UniMiB dataset contains human activity and fall data from 30 users (24 males and 6 females, aged 18 to 60). The data are divided into 9 types of daily activities and 8 types of falls. In this work, we utilize only daily activity data, which were collected under the guidance of predefined protocols. Participants were asked to perform each protocol twice, the first time with smartphone in the front right pocket and the second time in the front left pocket. The captured acceleration data were resampled with a sampling rate of 50 Hz and labeled with both activity and user labels. This dataset is publicly available and can be downloaded from [43].

REALDISP dataset consists of over 10 hours of activity data from 17 participants. The participants performed 33 activities, including whole body as well as body part specific activities, following the predefined protocols. The data were recorded by 9 IMUs distributed over the limbs and back of the participants. Each IMU provides acceleration, gyroscope, and magnetic field measurements, as well as orientation estimates. The recordings were sampled at 50 Hz. This dataset is publicly available and can be downloaded from [31].

Table 1. The details of the datasets used in our experiments.

	SBHAR	UniMiB	REALDISP
# of users	30	30	17
# of activities	12	9	33
Activities	walking, walking upstairs, walking downstairs, sitting, standing, laying, stand to sit, sit to stand, sit to lie, lie to sit, stand to lie, lie to stand	standing up from laying, lying down from standing, stand up from sitting, running, sitting down, go downstairs, going upstairs, walking, jumping	walking, jogging, running, jump up, jump front & back, jump sideways, jump leg/arms open/closed, jump rope, trunk twist (arms outstretched), trunk twist (elbows bended), waist bends forward, waist rotation, waist bends (reach foot with opposite hand), reach heels backwards, lateral bend, lateral bend arm up, repetitive forward stretching, upper trunk and lower body opposite twist, arms lateral elevation, arms frontal elevation, frontal hand claps, arms frontal crossing, shoulders high amplitude rotation, shoulders low amplitude rotation, arms inner rotation, knees (alternatively) to the breast, heels (alternatively) to the backside, knees bending (crouching), knees (alternatively) bend forward, rotation on the knees, rowing, elliptic bike, cycling
Sensors	1 3D accelerometer 1 3D gyroscope	1 3D accelerometer	9 IMUs
Frequency	50 Hz	50 Hz	50 Hz

The collected sensor data are typically sequences of frames acquired at successive equally spaced points in time. Each frame represents the values returned by all sensors at a point in time, and has both activity and

user labels. We segment sensor data into a set of equal-sized time windows using the sliding window method that is commonly used in AR and UR studies. The frame label that appears most in a time window is identified as the label of the time window. In our experiments, the sliding window with 2 seconds size and 50% overlap is used, and the time windows with missing values are removed. The details of the datasets used in our experiments are summarized in Table 1.

4.2 Experimental Setup

We implement METIER using TensorFlow, and the code is released on GitHub[1]. The weight factor α is set to 1. In the training process, Adam algorithm is used to optimize the network parameters. The learning rate l is dynamically controlled by a decreasing function, which is given formally by Equation 16:

$$l = l_{\min} + (l_{\max} - l_{\min}) \times e^{-\frac{i}{\mathrm{rs}}} \tag{16}$$

where $l_{\min}$ and $l_{\max}$ are minimum and maximum learning rates, respectively, which are set to 0.0005 and 0.003 in our experiments. rs controls the reduce speed of l and is set to 2000. i denotes the iteration step. The mini-batch size is set to 100.

Experiments are conducted on all above three datasets, and nested cross-validation is used for model evaluation. Nested cross-validation contains an outer loop of 10 folds and an inner loop of 5 folds. The original dataset is randomly split into 10 equal-sized sets. A set is used as test set, and the remaining 9 sets are combined as outer training set. This is repeated 10 times, with each of the 10 sets used exactly once as test set. Each outer training set is further sub-divided into 5 sets. A set is selected as validation set and the other 4 sets are combined into the corresponding inner training set. This is repeated 5 times, with each of the 5 sets used exactly once as validation set. The inner cross-validation is used for hyper-parameter optimization, and the outer for performance evaluation. Note that AR and UR models share the same training, validation, and test sets.

In our experiments, both accuracy and Macro F1-score (abbreviated as F1-score) are introduced as performance metrics. F1-score considers both precision and recall, which is given formally by Equation 17:

$$\mathrm{F1-score} = \frac{1}{C} \times \sum_{c=1}^{C} \frac{2 \times precision_c \times recall_c}{precision_c + recall_c} \tag{17}$$

where c is the class index and C denotes the number of classes.

4.3 Parameter Evaluation

To build a reliable model, we further investigate the impact of two key hyper-parameters, i.e., the number of convolutional layers and the dimension of state in bi-LSTM.

Convolutional layers are responsible for feature extraction and critical to model performance. We increase the number of convolutional layers in METIER from 1 to 5 and record the performance on SBHAR dataset. Results are shown in Fig. 4. As the number of convolutional layers increases, UR performance gradually declines, and AR performance increases first and then decreases. The best performance occurs when the number of convolutional layers is 2. The results suggest that continued increase in the number of convolutional layers will degrade model performance, probably because too many convolutional layers make it difficult to train the model.

The dimension of state is important for the representation ability of bi-LSTM. We increase the dimension from 32 to 256 with an increment step of 32 and record the performance on SBHAR dataset. Results are shown in Fig. 5. When the dimension increases from 32 to 64, the performance of these two tasks is significantly improved, which might be because the increase in dimension gives our model a stronger representation ability. As the dimension increases from 64 to 256, the performance of these two tasks changes slowly. AR performance increases first and then decreases, and UR performance slowly decreases. The continued increase

[1] https://github.com/yizhangzc/METIER/

in dimension does not lead to a continued increase in performance, which might be because the dimension is already able to meet the needs of the current tasks, and excessive parameters are prone to overfitting.

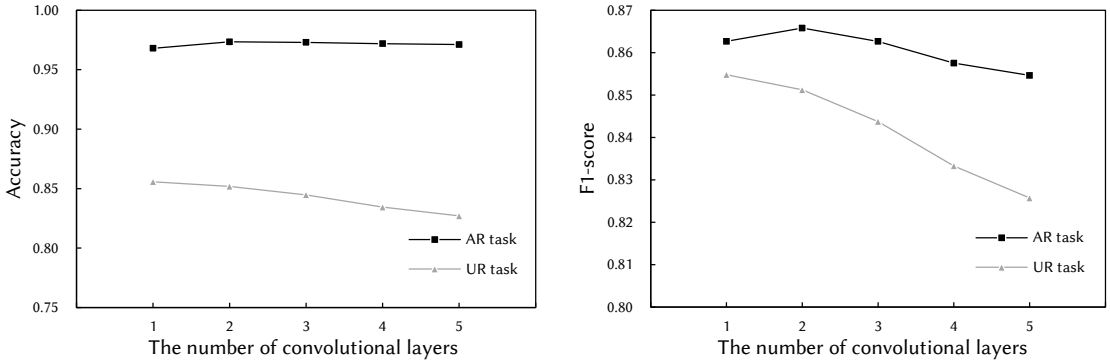

Fig. 4. The performance of AR and UR using different numbers of convolutional layers.

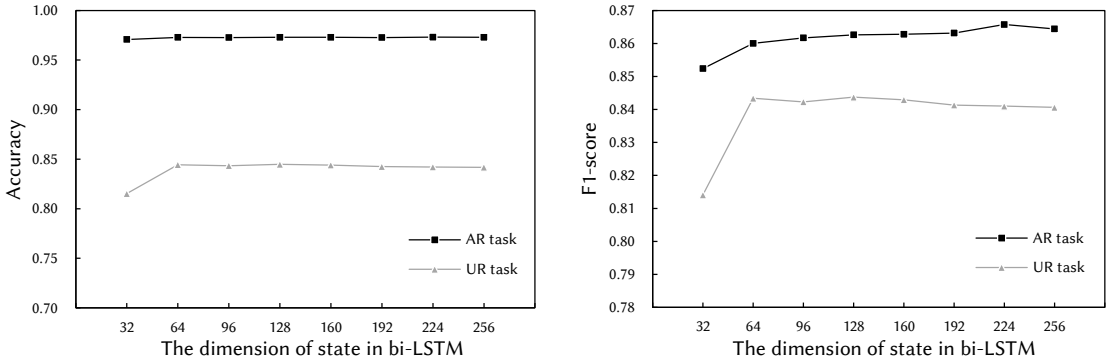

Fig. 5. The performance of AR and UR using different dimensions of state in bi-LSTM.

4.4 Comparison with Simplified Models

To demonstrate the effectiveness of soft parameter sharing and mutual attention mechanism, we compare METIER with its simplified models on two tasks. For the sake of fairness, all simplified models follow the same experimental setup as METIER, and their parameters are also optimized.

Simplified model:

METIER-STL: METIER-STL is the single-task learning version of METIER. Apart from the absence of soft parameter sharing and mutual attention mechanism, it shares the same architecture design with METIER.

METIER-SA: METIER-SA adds self-attention to METIER-STL, and the rest is the same as METIER-STL.

METIER-MA: METIER-MA removes soft parameter sharing from METIER, and the rest is the same as METIER.

METIER-SS: METIER-SS removes mutual attention mechanism from METIER, and the rest is the same as METIER.

In order to statistically measure the significance of performance differences, pairwise t-tests at 95% significance level are conducted between METIER and simplified models. The results are shown in Tables 2 and 3, which show the following tendencies:

(1) On both tasks, METIER-SA outperforms METIER-STL, and METIER-MA outperforms METIER-SA. The results demonstrate the effectiveness of attention mechanism. Compared to self-attention, mutual attention enables AR and UR tasks to exploit their knowledge to highlight important features for each other, and brings better performance on both AR and UR tasks.

(2) METIER-SS outperforms METIER-STL on both tasks, which suggests that by softly sharing parameters between AR and UR networks, the commonalities and differences across tasks can be exploited to promote these two tasks together.

(3) METIER is superior to METIER-SS and METIER-MA on both tasks, which implies that combining soft parameter sharing and mutual attention mechanism can achieve better performance. The results demonstrate the superiority of our model.

Table 2. The comparison between METIER and simplified models on AR task (mean±std). * means that METIER is statistically superior to the compared model (pairwise t-test at 95% significance level).

	SBHAR		UniMiB		REALDISP	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
METIER-STL	0.965±0.003*	0.863±0.015*	0.918±0.008*	0.903±0.009*	0.931±0.004*	0.948±0.002*
METIER-SA	0.968±0.002*	0.866±0.011*	0.924±0.009*	0.911±0.009*	0.932±0.004*	0.949±0.002*
METIER-MA	0.971±0.003*	0.877±0.017	0.929±0.007*	0.916±0.007*	0.934±0.003*	0.951±0.002*
METIER-SS	0.972±0.004*	0.870±0.018*	0.928±0.006*	0.913±0.005*	0.934±0.003*	0.952±0.002*
METIER (proposed)	**0.979±0.002**	**0.882±0.012**	**0.936±0.007**	**0.923±0.008**	**0.940±0.003**	**0.957±0.002**

Table 3. The comparison between METIER and simplified models on UR task (mean±std). * means that METIER is statistically superior to the compared model (pairwise t-test at 95% significance level).

	SBHAR		UniMiB		REALDISP	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
METIER-STL	0.847±0.008*	0.845±0.008*	0.809±0.011*	0.819±0.010*	0.952±0.003*	0.951±0.002*
METIER-SA	0.863±0.007*	0.861±0.007*	0.841±0.009	0.847±0.008*	0.953±0.003*	0.953±0.002*
METIER-MA	0.869±0.008*	0.868±0.008*	0.850±0.010	0.858±0.010	0.955±0.003*	0.954±0.002*
METIER-SS	0.860±0.010*	0.859±0.009*	0.829±0.013*	0.837±0.011*	0.955±0.003*	0.954±0.002*
METIER (proposed)	**0.876±0.006**	**0.875±0.007**	**0.857±0.008**	**0.865±0.005**	**0.960±0.003**	**0.960±0.002**

4.5 Comparison with Other AR and UR Models

To demonstrate the competitive performance of METIER, we compare it with the state-of-the-art AR and UR models. For the sake of fairness, all compared models follow the same experimental setup as METIER, and their parameters are also optimized.

AR models:

CNN15 [46]: CNN15 adopts a CNN architecture, consisting of three convolutional layers and two pooling layers, to automatically learn features from the raw inputs, and classifies activities by a softmax classifier.

DeepConvLSTM [27]: DeepConvLSTM combines CNN and LSTM into a unified deep architecture for AR. Four convolutional layers first extract features from raw sensor data, and then two LSTM layers learn temporal dependencies in features. Finally, a softmax classifier recognizes activities based on the outputs of LSTM. There is no pooling operation in DeepConvLSTM.

LSTM-F [13]: LSTM-F exploits three LSTM layers to process raw sensor data directly, and then classifies activities by a softmax classifier.

b-LSTM-S [13]: b-LSTM-S uses one bi-LSTM layer to learn temporal dependencies in raw sensor data from two directions, and classifies activities by a softmax classifier.

DeepSense [48]: DeepSense integrates CNN and GRU for time-series mobile sensing data processing. It splits a time window into a series of non-overlapping time intervals along time, and the frequency representations of each time interval are fed into a CNN, which consists of several individual three-layer convolutional subnets for the data from different sensors and a global three-layer convolutional subnet for the outputs of all individual convolutional subnets. The intra-interval representations are then fed into a two-layer GRU, and a following softmax classifier is used for classification.

UANN [17]: UANN performs a regular AR task and an adversarial UR task jointly. The network consists of a feature extractor, an activity classifier, and a user classifier. The feature extractor learns deep features from sensor data, and feeds them into activity and user classifiers. Regular AR task updates feature extractor and activity classifier to minimize activity classification loss. Adversarial UR task updates feature extractor to maximize user classification loss and updates user classifier to minimize user classification loss. Specifically, the feature extractor is a CNN consisting of three convolutional layers, three pooling layers, and a fully-connected layer. Activity classifier is a softmax classifier, and user classifier is a logistic regressor.

UR models:

DeepSense [48]: DeepSense is also used as a UR model, which has the same architecture as that described above.

UANN [17]: UANN is also used as a UR model, where AR task is adversarial and UR task is regular.

kNN18 [9]: kNN18 works on hand-crafted features, including mean, standard deviation, mean square root, mean absolute difference from the mean of data channels, magnitude, percentile 25, and percentile 75, which classifies users by a kNN classifier.

IDNet [10]: IDNet detects gait cycles from raw sensor data, and then feeds them into a CNN consisting of two convolutional layers and one pooling layer for feature extraction. Finally, a softmax classifier is used for classification.

DCNN [6]: DCNN transforms detected gait cycles into a 2D spectro-temporal space and builds a DNN consisting of five convolutional layers, three pooling layers, and one fully connected layer to extract deep features from the time-frequency expansions of gait cycles. A softmax classifier identifies users based on the outputs of the DNN finally.

Different from other models, IDNet and DCNN identify users based on gait cycles. We follow their data processing workflow to detect and extract gait cycles from raw sensor data, and use them as input. The rest of the models use the same input as METIER. In order to statistically measure the significance of performance differences, pairwise t-tests at 95% significance level are conducted between METIER and other models. The results are shown in Tables 4 and 5, which show the following tendencies:

(1) For AR task, METIER outperforms all other AR models. The accuracy and F1-score of METIER are 0.011 and 0.006 (on SBHAR dataset)/0.019 and 0.020 (on UniMiB dataset)/0.008 and 0.007 (on REALDISP dataset) higher than the best values of other AR models. The results demonstrate the superiority of our model, and suggest that by introducing user-related knowledge, our model can provide better AR performance.

(2) For UR task, METIER is superior to all other UR models. The accuracy and F1-score of METIER are 0.062 and 0.063 (on SBHAR dataset)/0.171 and 0.170 (on UniMiB dataset)/0.008 and 0.010 (on REALDISP dataset) higher than the best values of other UR models. The experimental datasets contain multiple activity scenarios, and the results show the superiority of our model in dealing with activity changes and indicate that activity-related knowledge can help our model work well in different activity scenarios.

(3) Compared to the AR models using CNN or RNN alone (i.e., CNN15, LSTM-F, and b-LSTM-S), the AR models combining CNN and RNN (i.e., DeepConvLSTM and METIER) provide stable performance on different datasets. It indicates that combining CNN and RNN can lead to better model robustness, and also demonstrates the superiority of our model architecture.

(4) UANN is the latest deep multi-task learning model that considers both AR and UR tasks. The comparison between UANN and METIER suggests that our model can better utilize the commonalities and differences across these two tasks to promote them simultaneously. METIER trains AR and UR networks

jointly, and performs two tasks simultaneously. UANN performs only one regular task, and adversarial task is used to help training. The adversarial training can increase model generalization, but also weaken the discriminative power of the features. This may be the reason why METIER outperforms UANN.

(5) IDNet and DCNN get poor performance. The main reason might be that their cycle detection is designed for walking and cannot work well with other activities. Extending the supported activity scenarios of UR is a non-trivial task, and our model can avoid this problem.

(6) The performance of DeepSense varies greatly across different datasets and tasks. The main reason might be because it splits a time window into a series of non-overlapping time intervals along time and only considers the temporal dependencies between time intervals. This architecture excels at learning long-term dependencies in sensor data, but ignores short-term dependencies.

Table 4. The comparison between AR models (mean±std). * means that METIER is statistically superior to the compared model (pairwise t-test at 95% significance level).

	SBHAR		UniMiB		REALDISP	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
CNN15	0.959±0.003*	0.841±0.011*	0.828±0.016*	0.793±0.015*	0.932±0.003*	0.950±0.002*
DeepConvLSTM	0.968±0.003*	0.863±0.013*	0.917±0.007*	0.903±0.007*	0.932±0.003*	0.948±0.002*
LSTM-F	0.966±0.003*	0.865±0.015*	0.896±0.011*	0.877±0.015*	0.929±0.003*	0.945±0.002*
b-LSTM-S	0.966±0.003*	0.876±0.014	0.872±0.020*	0.848±0.024*	0.929±0.003*	0.943±0.003*
DeepSense	0.966±0.003*	0.860±0.017*	0.870±0.008*	0.843±0.008*	0.932±0.003*	0.949±0.002*
UANN	0.958±0.003*	0.834±0.009*	0.862±0.006*	0.834±0.006*	0.930±0.003*	0.947±0.002*
METIER (proposed)	**0.979±0.002**	**0.882±0.012**	**0.936±0.007**	**0.923±0.008**	**0.940±0.003**	**0.957±0.002**

Table 5. The comparison between UR models (mean±std). * means that METIER is statistically superior to the compared model (pairwise t-test at 95% significance level).

	SBHAR		UniMiB		REALDISP	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
DeepSense	0.814±0.012*	0.812±0.012*	0.659±0.013*	0.668±0.013*	0.952±0.002*	0.950±0.002*
UANN	0.775±0.014*	0.772±0.013*	0.686±0.014*	0.695±0.013*	0.952±0.003*	0.950±0.001*
kNN18	0.770±0.006*	0.767±0.006*	0.674±0.011*	0.684±0.013*	0.950±0.002*	0.946±0.003*
IDNet	0.599±0.006*	0.603±0.005*	0.590±0.006*	0.611±0.003*	0.909±0.000*	0.901±0.000*
DCNN	0.470±0.004*	0.512±0.003*	0.505±0.002*	0.522±0.002*	0.798±0.010*	0.759±0.010*
METIER (proposed)	**0.876±0.006**	**0.875±0.007**	**0.857±0.008**	**0.865±0.005**	**0.960±0.003**	**0.960±0.002**

Table 6. The mean inference time of AR and UR models in SBHAR dataset (ms).

	METIER	CNN15	DeepConvLSTM	LSTM-F	b-LSTM-S	DeepSense	UANN	kNN18	IDNet	DCNN
AR task	10. 047	2.516	56.557	88.249	74.462	9.796	3.062			
UR task	10. 047					9.713	3.314	0.619	2.110	3.204

To demonstrate the computational efficiency of METIER, we compare the inference time between METIER and other AR and UR models. The inference is conducted on a GPU (NVIDIA RTX 2080Ti), and the mean inference time of 100 samples in SBHAR dataset is shown in Table 6. METIER takes about 10 ms to perform inference. Compared to the models using RNN (i.e., DeepConvLSTM, LSTM-F, b-LSTM-S, and DeepSense), it shows lower inference cost. However, its inference cost is significantly higher than the models using CNN (i.e.,

CNN15, UANN, IDNet, and DCNN) and traditional machine learning (i.e., kNN18). This is mainly because RNN significantly increases computation time.

5 CONCLUSIONS AND FUTURE WORK

AR and UR using wearable sensors are two key tasks in ubiquitous and mobile computing. In this paper, we propose METIER model to solve these two tasks jointly. It softly shares parameters between AR and UR networks, and optimizes these two networks jointly. The commonalities and differences across these two tasks are exploited to promote them together. Furthermore, mutual attention mechanism is introduced to enable AR and UR tasks to use their knowledge to highlight important features for each other. Experimental results demonstrate the effectiveness of METIER on both tasks.

In the future, we will extend our work from the following two directions. First, we will explore new parameter sharing methods, e.g., cross-stitch network, which can search for the best way to share automatically. Second, we will introduce more related tasks, e.g., sensor direction recognition and sensor placement recognition, and investigate their impact on model performance.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China under Grant No. 2018YFB0505000.

REFERENCES

[1] Oresti Baños, Miguel Damas, Héctor Pomares, Ignacio Rojas, Máté Attila Tóth, and Oliver Amft. 2012. A benchmark dataset to evaluate sensor displacement in activity recognition. In *Proceedings of the 14th ACM Conference on Ubiquitous Computing*. ACM, 1026–1035.

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

[3] Rich Caruana. 1997. Multitask learning. *Machine Learning* 28, 1 (1997), 41–75.

[4] Sangil Choi, Ik-Hyun Youn, Richelle LeMay, Scott Burns, and Jong-Hoon Youn. 2014. Biometric gait recognition based on wireless acceleration sensor using k-nearest neighbor classification. In *Proceedings of the 3rd International Conference on Computing, Networking and Communications*. IEEE, 1091–1095.

[5] James E Cutting and Lynn T Kozlowski. 1977. Recognizing friends by their walk: Gait perception without familiarity cues. *Bulletin of the Psychonomic Society* 9, 5 (1977), 353–356.

[6] Omid Dehzangi, Mojtaba Taherisadr, and Raghvendar ChangalVala. 2017. IMU-based gait recognition using convolutional neural networks and multi-sensor fusion. *Sensors* 17, 12 (2017), 2735.

[7] Gonzalo Bailador Del Pozo, Carmen Sánchez Ávila, Alberto de Santos Sierra, and Javier Guerra Casanova. 2012. Speed-independent gait identification for mobile devices. *International Journal of Pattern Recognition and Artificial Intelligence* 26, 8 (2012), 126001301–126001313.

[8] Mohammad Omar Derawi, Claudia Nickel, Patrick Bours, and Christoph Busch. 2010. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *Proceedings of the 6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, 306–311.

[9] Seham Abd Elkader, Michael Barlow, and Erandi Lakshika. 2018. Wearable sensors for recognizing individuals undertaking daily activities. In *Proceedings of the 22nd ACM International Symposium on Wearable Computers*. ACM, 64–67.

[10] Matteo Gadaleta and Michele Rossi. 2018. IDNet: Smartphone-based gait recognition with convolutional neural networks. *Pattern Recognition* 74 (2018), 25–37.

[11] Haodong Guo, Ling Chen, Liangying Peng, and Gencai Chen. 2016. Wearable sensor based multimodal human activity recognition exploiting the diversity of classifier ensemble. In *Proceedings of the 18th ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 1112–1123.

[12] Haodong Guo, Ling Chen, Yanbin Shen, and Gencai Chen. 2014. Activity recognition exploiting classifier level fusion of acceleration and physiological signals. In *Proceedings of the 16th ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 63–66.

[13] Nils Y Hammerla, Shane Halloran, and Thomas Plötz. 2016. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. AAAI, 1533–1540.

[14] Jin-Hyuk Hong, Julian Ramos, and Anind K Dey. 2016. Toward personalized activity recognition systems with a semipopulation approach. *IEEE Transactions on Human-Machine Systems* 46, 1 (2016), 101–112.

[15] Yu-Jin Hong, Ig-Jae Kim, Sang Chul Ahn, and Hyoung-Gon Kim. 2010. Mobile health monitoring system based on activity recognition using accelerometer. *Simulation Modelling Practice and Theory* 18, 4 (2010), 446–455.

[16] Verne Thompson Inman, Henry James Ralston, and Frank Todd. 1981. *Human Walking*. Williams & Wilkins.

[17] Yusuke Iwasawa, Kotaro Nakayama, Ikuko Eguchi Yairi, and Yutaka Matsuo. 2017. Privacy issues regarding the application of DNNs to activity-recognition using wearables and its countermeasures by use of adversarial training. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI, 1930–1936.
[18] Aftab Khan, Sebastian Mellor, Eugen Berlin, Robin Thompson, Roisin McNaney, Patrick Olivier, and Thomas Plötz. 2015. Beyond activity recognition: Skill assessment from accelerometer data. In *Proceedings of the 17th ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 1155–1166.
[19] Md Abdullah Al Hafiz Khan, Nirmalya Roy, and Archan Misra. 2018. Scaling human activity recognition via deep learning-based domain adaptation. In *Proceedings of the 16th IEEE International Conference on Pervasive Computing and Communications*. IEEE, 1–9.
[20] Oscar 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.
[21] Shinya Matsui, Nakamasa Inoue, Yuko Akagi, Goshu Nagino, and Koichi Shinoda. 2017. User adaptation of convolutional neural network for human activity recognition. In *Proceedings of the 25th European Signal Processing Conference*. IEEE, 753–757.
[22] Daniela Micucci, Marco Mobilio, and Paolo Napoletano. 2017. Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences* 7, 10 (2017), 1101.
[23] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent models of visual attention. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. MIT Press, 2204–2212.
[24] Sebastian Münzner, Philip Schmidt, Attila Reiss, Michael Hanselmann, Rainer Stiefelhagen, and Robert Dürichen. 2017. CNN-based sensor fusion techniques for multimodal human activity recognition. In *Proceedings of the 21st ACM International Symposium on Wearable Computers*. ACM, 158–165.
[25] Claudia Nickel, Christoph Busch, Sathyanarayanan Rangarajan, and Manuel Möbius. 2011. Using hidden markov models for accelerometer-based biometric gait recognition. In *Proceedings of the 7th IEEE International Colloquium on Signal Processing and Its Applications*. IEEE, 58–63.
[26] Guillaume Obozinski, Ben Taskar, and Michael Jordan. 2006. Multi-task feature selection. *Statistics Department, UC Berkeley, Technical Report* 2 (2006).
[27] Francisco Javier Ordóñez and Daniel Roggen. 2016. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16, 1 (2016), 1–25.
[28] Ioannis Papavasileiou, Savanna Smith, Jinbo Bi, and Song Han. 2017. Gait-based continuous authentication using multimodal learning. In *Proceedings of the 2nd IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies*. IEEE, 290–291.
[29] 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.
[30] 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. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 2 (2018), 74.
[31] REALDISP dataset. Available online: https://archive.ics.uci.edu/ml/datasets/REALDISP+Activity+Recognition+Dataset
[32] Attila Reiss and Didier Stricker. 2013. Personalized mobile physical activity recognition. In *Proceedings of the 17th Annual International Symposium on Wearable Computers*. ACM, 25–28.
[33] Jorge-L Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, and Davide Anguita. 2016. Transition-aware human activity recognition using smartphones. *Neurocomputing* 171 (2016), 754–767.
[34] Seyed Ali Rokni, Marjan Nourollahi, and Hassan Ghasemzadeh. 2018. Personalized human activity recognition using convolutional neural networks. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. AAAI, 8143–8144.
[35] Liu Rong, Duan Zhiguo, Zhou Jianzhong, and Liu Ming. 2007. Identification of individual walking patterns using gait acceleration. In *Proceedings of the 1st International Conference on Bioinformatics and Biomedical Engineering*. IEEE, 543–546.
[36] Theresia Ratih Dewi Saputri, Adil Mehmood Khan, and Seok-Won Lee. 2014. User-independent activity recognition via three-stage GA-based feature selection. *International Journal of Distributed Sensor Networks* 10, 3 (2014), 706287.
[37] SBHAR dataset. Available online: http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions
[38] Pekka Siirtola and Juha Röning. 2012. Recognizing human activities user-independently on smartphones based on accelerometer data. *International Journal of Interactive Multimedia and Artificial Intelligence* 1, 5 (2012), 38–45.
[39] Sebastijan Sprager and Damjan Zazula. 2009. A cumulant-based method for gait identification using accelerometer data with principal component analysis and support vector machine. *WSEAS Transactions on Signal Processing* 5, 11 (2009), 369–378.
[40] 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 11th IEEE International Conference on Data Mining*. IEEE, 1218–1223.
[41] 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.
[42] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. ACL, 1722–1731.
[43] UniMiB dataset. Available online: https://www.sal.disco.unimib.it/technologies/unimib-shar/
[44] Yunus Emre Ustev, Ozlem Durmaz Incel, and Cem Ersoy. 2013. User, device and orientation independent human activity recognition on mobile phones: Challenges and a proposal. In *Proceedings of the 15th ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. ACM, 1427–1436.
[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran

Associates Inc., 6000–6010.
[46] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. 2015. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI, 3995–4001.
[47] Yongxin Yang and Timothy M. Hospedales. 2017. Deep multi-task representation learning: A tensor factorisation approach. In *Proceedings of the 5th International Conference on Learning Representations*.
[48] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. 2017. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web*. ACM, 351–360.
[49] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. 2014. Facial landmark detection by deep multi-task learning. In *Proceedings of the 13th European Conference on Computer Vision*. Springer, 94–108.
[50] Zhongtang Zhao, Yiqiang Chen, Junfa Liu, Zhiqi Shen, and Mingjie Liu. 2011. Cross-people mobile-phone based activity recognition. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. AAAI, 2545–2550.