LEARNING EMBEDDINGS FOR WEARABLE-BASED HUMAN ACTIVITY ANALYSIS

by

TAORAN SHENG

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
August 2020

To my family

For their love, support, care and patience

ACKNOWLEDGEMENTS

I would like to express the sincere appreciation to my supervising professor Dr. Manfred Huber, who guides me throughout the development of my research with great patience. Without his inspiration and helps, this would not have been possible. I also would like to express my gratitude to the committee members: Dr. Vassilis Athitsos, Dr. Farhad Kamangar, and Dr. Gergely Zaruba, for their valuable suggestions and insightful comments regarding my research work.

Furthermore, I would like to thank my labmates in Learning and Adaptive Robotics (LEARN) lab. It is a great pleasure to have this opportunity to work closely with all of them. Finally, I am grateful for my family. The completion of this work would not have been possible without them. Thank you for everything you have done for me.

August 24, 2020

ABSTRACT

LEARNING EMBEDDINGS FOR WEARABLE-BASED HUMAN ACTIVITY ANALYSIS

TAORAN SHENG, Ph.D.

The University of Texas at Arlington, 2020

Supervising Professor: Manfred Huber

The embedded sensors in widely used smartphones, wearable devices and smart environments make the sensor data stream of human activity more accessible. With the development of deep neural networks, extensive studies have been conducted using deep learning methods to extract useful information from the sensor data to recognize the human activity, identify the person, or monitor the health condition of the person. However, applying deep neural networks to the sensor based human activity analysis task remains a challenging research problem in ubiquitous computing. Some of the reasons are: (*i*) The majority of the acquired data has no labels; (*ii*) Most of the previous works in activity and sensor stream analysis have been focusing on one aspect of the data, e.g. only recognizing the type of the activity or only identifying the person who performed the activity; (*iii*) Segmenting a continuous sensor stream and preserving the completeness of each human activity is difficult. In this dissertation, various deep learning techniques have been studied to address these problems in a weakly supervised, unsupervised, or semi-supervised manner. All the developed techniques use deep learning networks to learn embedding spaces in which

activities group and thus classifiers can be trained efficiently. For this, both siamese network architectures for weakly supervised data and autoencoder-type networks for unsupervised techniques are learned and combined.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

With the development and increased availability of embedded sensors in smart environments and wearable devices, a vast potential to improve the quality of human life has emerged in many different areas, such as: smart assistive technologies, human computer interaction, health management, etc. These areas can benefit from the information extracted from the sensor stream. While many current machine learning methods have been studied in these areas and achieved impressive results, most of the existing systems still suffer from the following limitations: (i) The sliding window is used to segment the sensor stream [1, 2]. This segmentation method is simple, yet, it can segment a complete activity into pieces and destroy the completeness of the activity. In addition, designing the window requires domain knowledge to determine the window size and the sliding speed. (ii) Existing methods rely heavily on labeled data to guide the learning of the model [3, 4]. However, acquiring a huge amount of labeled data is very difficult and the majority of the accessible data has no labels. (iii) The previous methods intend to extract useful information from one aspect of the sensor data. For example, some systems only recognize the type of the activity [5, 6], some systems only identify the person who performed the activity [7, 8]. Very few works have been done on solving these related tasks together. (iv) Most of the current person identification systems depend on face, fingerprint, iris, or gait etc. These identification methods are effective, but require specific input from the users [9, 10], while the sensor based system can provide an unobtrusive and convenient way to complete the task.

This work presents approaches that attempt to mitigate these limitations. Firstly, the proposed models are based on the siamese network and autoencoder architectures, which enable the weakly supervised and unsupervised learning of the model, hence reducing the needs for labeled data. Secondly, automatic segmentation of the sensor stream has also been achieved by using the siamese architecture and the side information of the data. Thirdly, it is intuitive that different persons perform the activities in different ways, hence, distinct personal characteristics are commonly present in all the activities performed by the person. Based on this observation, it seems natural to address the activity recognition problem and the person identification problem together. Thus, a multi-tasking model is proposed, which can solve multiple related tasks (e.g. activity recognition, person identification etc.) using the sensor stream at the same time. Fourthly, inspired by the physical property in the real world that the objects have inertia and their states usually change slowly and infrequently, an autoencoder based model is proposed to learn activity categories in a completely unsupervised way. This autoencoder learns to retain the slow features [11] in the activity data, which are more relevant to the activity type, and to disregard the fast features, which relatively irrelevant to the activity type. Therefore the learned model is more effective on retaining the activity type information, can provide more meaningful representations to the subsequent activity clustering algorithm, and improves the performance on the activity clustering task. In addition, this dissertation also introduces a model that utilize an unsupervised learned embedding space for primitive activities to build a hierarchical model for activity learning and recognition that facilitates zero-shot and few-shot learning of novel activities form very limited observations for this model. Finally, combining principles and techniques for the supervised and unsupervised learning approaches developed here, weakly semi-supervised techniques

are developed to allow for improved classification using limited weakly supervised data while utilizing a larger set of unsupervised data.

In the remainder of this dissertation, each chapter introduces one of the developed models and discusses the corresponding related work. In particular, Chapter 2 presents a model weakly supervised human activity recognition and segmentation using a siamese network architecture [12]. Chapter 3 expands this architecture to address multi-task representation learning [13]. A novel unsupervised representation learning approach for HAR [14] is then introduced in Chapter 4, before Chapter 5 introduces a weakly semi-supervised technique that combines principles form the weakly-supervised and unsupervised models to allow representation learning using small amounts of labeled data with larger amounts of unlabeled data for higher accuracy learning. Chapter 6, finally, introduces a hierarchical activity modeling approach for zero-shot and few-shot learning of new activities. Chapter 7 then concludes the dissertation.

CHAPTER 2

Weakly Supervised Human Activity Segmentation and Recognition

2.1 Introduction

Human Activity Recognition (HAR) is critical in many research areas, including human computer interaction, smart assistive technologies etc. These areas use HAR systems to provide information about people's activities and behaviors. The common way to implement such a system is by collecting data from environmental or wearable sensors and processing this data with machine learning algorithms. One of the dominant framework types [15] used in HAR systems relies on (*i*) sliding window segmentation of time series data recorded by wearable sensors, (*ii*) manually designed features, such as statistical mean, variance, entropy of the signal, and features extracted from the frequency domain, and (*iii*) different supervised classification algorithms to recognize activity. This type of framework performs well; however, domain knowledge is required to determine the window size, sliding speed, and to design the features manually. Recently, many deep neural networks (DNN) have been developed and applied in HAR systems [3, 4, 5]. These methods can automatically extract features from the data without any domain knowledge and have been shown to be useful in HAR applications. But, they still require explicit labels to supervise the training of the model and usually a hand designed sliding window method to segment the time series data. Our work is motivated by two factors: enabling automatic segmentation of the time series and limiting the supervision required while learning a recognition model. In this paper we propose an approach based on multiple siamese networks to segment and recognize human activities in sensor data streams. The

proposed approach learns one siamese network to automatically segment the times series without manually designing a sliding window, and another siamese network to provide a similarity metric that can be used to cluster the activities without using explicitly labeled data.

2.2 Related Work

The typical process [16] of human activity recognition includes signal preprocessing, data segmentation, feature extraction, and applying machine learning to recognize each activity. In this work, we mainly focus on the data segmentation stage and the activity recognition stage.

2.2.1 Time Series Data Segmentation

The data segmentation stage identifies the segments in the data stream that are likely to be an activity and determines the start time t_{start} and end time t_{end} of the activity. Once all the activity segments in the data stream are identified, they can be fed into the recognition module.

Segmenting a continuous sensor stream is difficult. Because different activities continuously performed by people smoothly blur into each other, they are not clearly separated by a predefined posture or pause. One common approach used to segment the sequence data is using a sliding window [1, 2]. A window with predefined length and step size moves over the data stream, and the data sequence contained within the window is used as one data sample, which has one data label. This method, however, will introduce inaccuracy into the segmentation borders. A larger window size is needed to catch complex activities; but the larger the window, the less accurately the segmentation borders can be defined. Another segmentation method is based on signal energy [17]. It assumes that the intensities of different activities are different

and that the different intensities can be used as an indicator to determine the borders of an activity.

2.2.2 Feature Extraction and Recognition

In order to recognize an activity, discriminative features are needed. They can be designed with domain knowledge manually or learned by neural networks automatically. Hand designed features are widely used in HAR [15, 18]. They include statistical features, such as mean, variance and entropy, or features extracted in the frequency domain using a fourier transform, wavelet transform [19] or discrete cosine transform [20] etc. The advantage of these features is that they can be derived from the signal easily and have been shown to be effective in the HAR system.

With the advances of DNN, many HAR systems adopt DNN to allow automatic extraction of meaningful features. Zeng et al. [3] used convolutional neural networks (CNN) to capture local dependencies and identify scale invariant features of the activity signals. The local connectivity constraint between adjacent layers in the CNN forces the model to capture the local dependencies. Morales and Roggen [21] proposed models that combine CNN and Long Short Term Memory cells (LSTM) [22]. LSTM keeps track of an internal state that represents its memory. The memory state eases the learning of long time scale temporal dependencies and helps the HAR system to model more complex activities.

2.2.3 Siamese Neural Networks

A Siamese Network [23] is a neural network with two branches and tied weights. It processes two different inputs and yields two comparable representation vectors, which represent the features of each input, respectively. Then the representation vectors are fed deeper into the network, which consists of a predefined metric layer [24]

or a learned metric network to measure how similar the two inputs are. The siamese network is widely used to learn non-linear metrics, and has been successfully applied in computer vision, speech recognition, etc. A siamese CNN has been used to learn a complex similarity metric for face verification in [25]. Mueller and Thyagarajan [24] proposed a siamese RNN to measure semantic similarity between sentences. Zeghidour et al. [26] used a multi-output triamese network to identify the speaker and phonetic similarities jointly. Neculoiu et al. [27] used a siamese RNN for job title normalization.

2.3 Proposed Approach

Our proposed model contains two modules, a segmentation module and a recognition module. Both of them are using a similar siamese architecture (see Fig. 2.1) but with different LSTM structures and similarity functions.

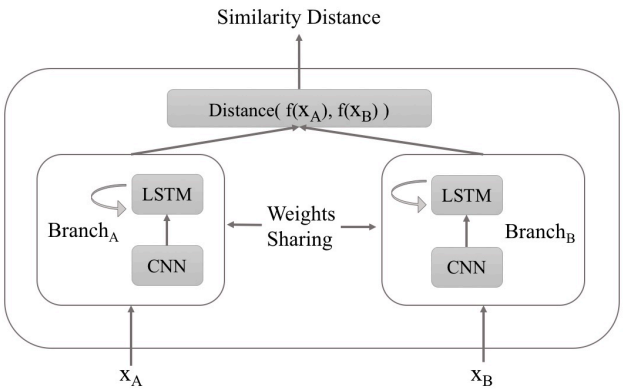


Figure 2.1: The basic siamese architecture used in our model

2.3.1 Siamese Architecture in the Proposed Approach

As shown in Fig. 2.1, given an input pair of human activity data sequences (x_A, x_B), the siamese network learns to map it to the representation space $(f(x_A), f(x_B)) \in R^d$. Then the layers above the dual-branch represent a similarity function that measures the distance between these two representation vectors. We will detail the siamese architecture in this section. The segmentation module and recognition module will be introduced in Subsections 2.3.2 and 2.3.3, respectively.

Our siamese networks share weights across their two branches. Each branch uses the same building blocks: (i) Dilated temporal convolutional layers, and (ii) Residual LSTM layers. Fully connected layers (FC) will be adopted to process the outputs from both branches. Fig. 2.2 illustrates details in each branch.

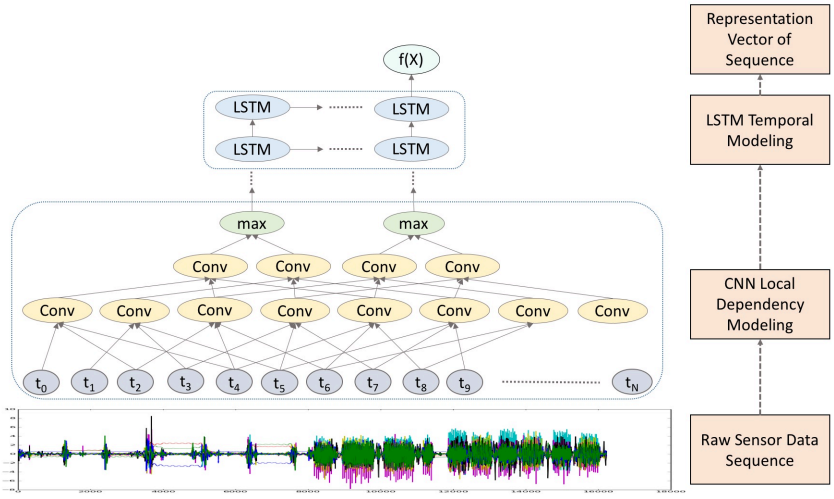


Figure 2.2: Details of a single branch in the siamese network

- *Dilated Temporal Convolutional Layer*

  The temporal convolutional layer is applied to the raw data sequences with the aim of matching the local pattern of the input data and enabling translational invariance of each pattern in the activity data sequence. To increase the

receptive field of the temporal convolutional layer, we adopt dilated temporal convolution, as it supports faster expanding receptive fields without losing resolution or coverage [28]. Batch Normalization (BN) [29] is applied after each temporal convolutional layer and before the non-linear activation function. BN can accelerate the learning process by preventing the internal covariate shift problem, allowing each intermediate layer not to have to adapt individually to a new distribution in every training step. In addition, BN provides a slight regularization effect to the network and can also prevent the early saturation of the non-linear activation function. To further reduce the temporal dimensions of the data sequence while introducing slight translational invariance in time, we insert a temporal max-pooling layer after every two dilated temporal convolutional layers. Max-pooling outputs the maximum within the region of a predefined pooling size and corresponds to a subsampling. The output of the max-pooling layer is fed into the subsequent LSTM layers.

- *Residual LSTM Layer*

  The LSTM layers are responsible for modeling the higher level temporal patterns in the data. The definition of an LSTM cell is as follows:

$$i_t = \sigma_g(W_i * x_t + U_i * h_{t-1} + b_i) \tag{2.1}$$

$$f_t = \sigma_g(W_f * x_t + U_f * h_{t-1} + b_f) \tag{2.2}$$

$$\tilde{c_t} = \sigma_c(W_c * x_t + U_c * h_{t-1} + b_c) \tag{2.3}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c_t} \tag{2.4}$$

$$o_t = \sigma_g(W_o * x_t + U_o * h_{t-1} + b_o) \tag{2.5}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{2.6}$$

A LSTM unit maintains a cell state c_t which contains the information from the past. Eq. (2.1) defines an input gate, which decides how much of the input x_t will be fed into the cell state. Eq. (2.2) defines a forget gate which decides to what level information will be removed from the cell state c_t. Eq. (2.3) defines the way to compute a cell state candidate $\tilde{c}_t$ that will be used to update the cell state. Then we can use the gates and the candidate to update the cell state as shown in Eq. (2.4). Eq. (2.5) defines an output gate. The output will be based on the cell state but will be a filtered version. The output gate works as the filter to decide what parts of the cell state will be the output, as shown in Eq. (2.6). It can be seen from the equations that the output of an LSTM at time step t only depends on x_t and h_{t-1}, which are the input at time step t and the information from the previous time steps. Thus, the information from the time steps after t is not used in the equations. To address this, we employ two different LSTM structures for the segmentation module and the recognition module, which will be covered in Subsections 2.3.2 and 2.3.3, respectively. Residual connections [30] are adopted between adjacent LSTM layers in our model, because the residual connection encourages the higher layer to learn something different from what the lower layer has already learned. It also encourages gradient flow.

- *FC Layer*

  The FC layers represent a sequence of non-linear transformations to the CNN-LSTM extracted features. Each FC layer corresponds to a linear transformation and a rectified-linear (ReLU) [31] activation function. Batch Normalization is used after each linear transformation and before the activation function, following [29]. The FC layer yields different outcomes for different modules. For the segmentation module it differentiates if the current frame belongs to the

boundaries of an activity. For the recognition module it works as a similarity metric that measures how similar the two inputs are.

Following the notation in [32], the shorthand description of each single branch is as follows: $C(64) - C(64) - P - C(64) - C(64) - P - R(128) - R(128)$, where $C(n)$ denotes a convolutional layer with n feature maps, P a max-pooling layer, and $R(n)$ a recurrent LSTM layer with n cells.

2.3.2 Segmentation Module

The periods that blur finishing and beginning two consecutive activities are transitions. They have features from both sides and contain the boundary of activities. The exact position of the boundary is often difficult to define in the data stream. But, we can define that the boundary is in the transition because the previous activity has finished within this period and the subsequent activity has started within this period. Thus, it is reasonable to either label the transition as an independent transition activity or as an unknown activity, or to include it into the activity before or after the transition period [15]. An unknown activity represents an infinite space of arbitrary activities. The space of transitions is not infinite, but if there are n different activities, there will be $\frac{n!}{(n-2)!}$ kinds of possible transitions that are all very short. These reasons make learning an explicit model for transition and unknown activity difficult. However, transitions and unknown activities occur between activities of interest; this characteristic allows transition and unknown activity to be used as the boundary period to the activity sequence. It is important to notice that if the transition and unknown activity are used as the boundary of an activity, then the segmentation is not a hard segmentation but rather a soft segmentation. The position of a soft segmentation is not a single frame but a period that contains one or more than one frame. To address this ambiguity of the transition period, multi-

ple segmentations and classifications of the resulting segments should be considered accurate representations of the data.

Table 2.1 shows the error assessment [15] used in this paper for judging correct segmentation and recognition. In the table, A, B, C are activities of interest, U is unknown activity, T is transition.

Table 2.1: Error assessment table.

Ground-Truth	**Segmentation & Recognition**	**Error Evaluation**
Basic Activities		
A-A-A	A-A-A	correct
A-A-A	A-B-A	incorrect
A-A-A	A-T-A	incorrect
A-A-A	A-U-A	incorrect
Without Transitions		
A-B	A-B	correct
A-B	A-C-B	incorrect
A-B	A-T-B	incorrect
A-B	A-U-B	incorrect
With Transitions		
A-T-B	A-B	correct
A-T-B	A-C-B	incorrect
A-T-B	A-T-B	correct
A-T-B	A-U-B	correct

Segmenting a data sequence with the proposed module can be formulated as follows: given the current time step t_M and an activity sequence $S =< a_0, \ldots, a_M, \ldots, a_N >$ from time step t_0 to t_N, $(0 < M < N)$, the proposed segmentation module predicts whether t_M is a segmentation frame in the data sequence. We consider the boundary as a frame or a list of consecutive frames that indicate that the activities before and after it are different. Thus, we split the sequence S at time step t_M into a pair of subsequences (sub_A, sub_B). Let $sub_A =< a_0, \ldots, a_M >$ be the activity sequence from time step t_0 to t_M, representing the history of information from previous time steps.

And let $sub_B =< a_N, \ldots, a_{M+1} >$ be the activity sequence from time step t_{M+1} to t_N in reverse chronological order, representing the future sequence that contains information from a future time step to the current time step. We restrict the length of the future sequence, so that the time delay in the system is not infinite.

As shown in Fig. 2.1, for the segmentation module $Branch_A$ is responsible to learn a representation vector $\vec{V}_{history}$ from the history sequence, while $Branch_B$ is responsible to learn a representation vector $\vec{V}_{future}$ from the future sequence. We assume here that if t_M is a segmentation frame, $\vec{V}_{history}$ and $\vec{V}_{future}$ should be far away from each other in the representation space; otherwise they should be close as they are then considered to be part of the same activity. In this way, the distance metric encoded in the representation space represents the transition process nicely, because the distance between the two vectors changes from short to long when the previous activity is finishing and the subsequent activity is starting.

The two branches in the segmentation siamese network use the information before and after the current time step explicitly and process the data sequence in opposite directions. The LSTM employed in each branch is a uni-directional LSTM, not a bi-directional LSTM (BLSTM). The reason for this is that [33] shows that in an LSTM autoencoder where the decoder decodes the target sequence in reverse order, it can help the model to capture the short range correlations easily. Thus, $Branch_A$ processing the history data in chronological order while $Branch_B$ processes the future data in reverse chronological order also aims to capture short range correlations because short range correlations can restrict the segmentation in a short region, while long range correlations can not.

The FC layers after the LSTM learn a non-linear distance metric to measure the similarity between $\vec{V}_{history}$ and $\vec{V}_{future}$, then output this distance. A generalized Gaussian function, $\mathcal{GG}(x;\beta) = \frac{\beta}{2\alpha\Gamma(1/\beta)} e^{-(|x-\mu|/\alpha)^\beta}$, is used as the distance target to

train the model. β here is a shape parameter of the function; if β is picked properly, the generalized Gaussian function can have a flat top and sharp edge that represents the occurrence and duration of a transition between activities. In the test stage, we use the segmentation module as a classifier. For an input frame, if the output from the segmentation module is above a threshold, we consider the input frame as a segmentation frame. Note that, since we adopt soft segmentation, it is possible that a list of consecutive frames can all be identified as segmentation positions.

2.3.3 Recognition Module

The recognition module sticks to the same siamese architecture described in Subsection 2.3.1, but the LSTM layers are different. In contrast to the segmentation network, a BLSTM is adopted here, which uses two uni-directional LSTMs working on the same data but in opposite directions along the time domain. The outputs from both uni-directional LSTMs are concatenated to be fed into the next layer of the network. The BLSTM structure aims to extract the information from the full sequence and balance the importance of both sides of the sequence. In comparison to the LSTM layers used in the segmentation module, the BLSTM used here learns to capture both short and long range correlations in the data sequence.

The recognition siamese network is trained with triplets (x_A, x_B, y), where x_A and x_B are the segmented sequences and $y \in \{0, 1\}$ indicates that x_A and x_B are the same kind of activity, $y = 1$, or different kinds of activities, $y = 0$.

The similarity function for the recognition siamese network is predefined as $D(x_A, x_B) = exp(-|f(x_A) - f(x_B)|) \in \{0, 1\}$ [24]. It forces the model to learn a mapping $f(x)$ that captures the critical similarities between the input pairs (x_A, x_B), and if $y = 1$, then $f(x_A), f(x_B) \in R^d$ should stay close, while $f(x_A), f(x_B)$ should be

far away from each other if $y = 0$. The mean-squared-error (MSE) is used to measure the loss between the model estimated similarity and the ground-truth similarity.

Assuming a list of segmented sequences: $l = \{act\_seq_0, \ldots, act\_seq_M, \ldots, act\_seq_N\}$, the trained recognition siamese network maps all the activity segments in l into the representation space. Then, different clustering algorithms can be used on those representation vectors to build the clusters. Here, single-linkage clustering is adopted. A drawback of single-linkage is that it tends to build long thin clusters in which the nearby elements of the same cluster have small distance values while the elements at the edge of a cluster may have larger distance to the elements from the opposite side of the same cluster than they have to the elements of other clusters [34]. This may be a problem in other research areas, but it aligns with some features of human activity sequences. Here, the data stream of activities consists of activities of interest interwoven with transitions. As discussed above, the exact borders of activities and transitions are difficult to define. The beginning and ending stages of a transition can be viewed as part of the previous and subsequent activities that it connects with. However, it should be noticed that transitions between different combinations of activities are essentially different, e.g. the transition from *lying* to *sitting* can not be the same as the transition from *walking* to *standing*. These transitions may have a large distance between each other, but are close to their connected activities, respectively. The single-linkage clustering preserves this feature naturally.

2.4 Evaluation and Experiment

2.4.1 Datasets

For empirical evaluation and comparison with other approaches, we test our method on three public datasets that contain the raw sensor data sequence of human activity. The descriptions of the datasets are listed below.

The **Daphnet Gait dataset (DG)** [35]: This dataset is collected from 10 patients with Parkinson's disease. Three triaxial acceleration sensors are fixed at the patient's ankle, upper leg, and trunk to record the activities performed by the patient with a sampling rate of 64Hz. The patients were instructed to carry out activities that are likely to induce freezing of gait (FoG). The FoG is a common symptom in Parkinson's disease that affects patient's activities like *walking*. The objective is to identify the FoG of the patients. We follow the same settings as in [4] and use the records of patient 9 for validation, the records of patient 2 for test, and the rest for training.

The **WISDM dataset** [18]: This dataset contains data collected from 36 users performing 6 different activities, including *jogging*, *ascending stairs*, etc., in a controlled experiment environment. The data is recorded with one triaxial accelerometer in a smartphone at a sampling rate of 20Hz.

The **SBHAR dataset** [36]: This dataset provides data from a group of 30 volunteers with an age bracket of 19-48 years, carrying out 6 basic activities, such as *walking* and *lying*, and 6 postural transitions such as $stand - sit$ and $sit - lie$. The data is recorded by letting the volunteers wear a smartphone on the waist during the experiment. The smartphone's embedded triaxial accelerometer and a gyroscope recorded the data at a sampling rate of 50Hz.

2.4.2 Performance Measures

In the experiments, we use two performance metrics: many-to-one accuracy and weighted F_1 score.

Because our model learned a metric without explicit labels, there is no straightforward way to compare the ground truth labels with the clusters found by an unsupervised algorithm using our learned distance metric. Here, we use many-to-one accuracy as a mapping-based performance measures to evaluate the performance. For the cluster with label t, select the most frequent correct label t^* for those frames in the cluster. Replace t with t^*. After processing each cluster this way, compute the accuracy as usual.

The weighted F_1 score has been used as the performance metric (for the DG dataset) in related work. In order to compare our results to the state-of-the-art we also calculate the weighted F_1 score:

$$F_w = 2 \sum_{i=1}^{C} \frac{N_i}{N_{total}} \frac{Precision_i \times Recall_i}{Precision_i + Recall_i} \tag{2.7}$$

where N_i is the number of samples in class i, N_{total} is the total number of samples, and for the given class i, $Precision_i = \frac{TP_i}{TP_i+FP_i}$, $Recall_i = \frac{TP_i}{TP_i+FN_i}$ and $i = 1, \ldots, C$ is the set of classes. TP_i, FP_i represent the number of true positives and false positives, FN_i is the number of false negatives.

2.4.3 Experimental Results

Tables 2.2, 2.3, and 2.4 illustrate the comparison of the proposed method against existing supervised and unsupervised methods on the DG, WISDM and SBHAR datasets.

Table 2.2: Results on the DG dataset in terms of weighted F_1 score.

Method	F_1 **Score**
LSTM baseline [37]	0.6675
DeepConvLSTM [37]	0.7344
LSTM-S [4]	0.7600
LSTM + Continuous Temporal, Sensor Attention [37]	0.8373
Proposed Method	**0.8952**

Table 2.3: Results on the WISDM dataset in terms of accuracy.

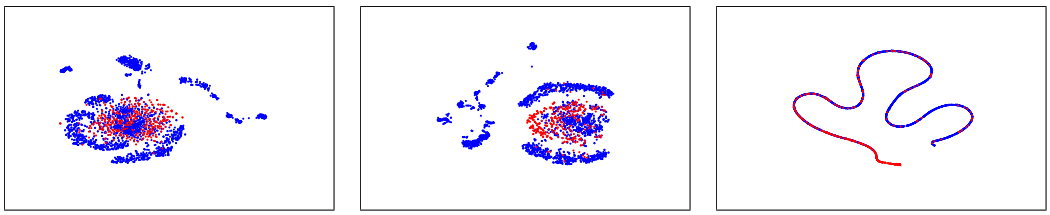
Method	**Accuracy(%)**
Multilayer Perceptron [18]	91.7
Ensemble Learning [38]	94.3
CNN with partial weight sharing [39]	96.88
DBN+HMM[40]	**98.23**
Proposed Method	96.68

The results in Table 2.2 show that the proposed weakly supervised method outperforms the state-of-the-art, LSTM with continuous temporal and sensor attention [37] , on the DG dataset. Tables 2.3, 2.4 show that the best performance on the WISDM and SBHAR datasets are achieved by supervised methods, DBN+HMM [40] and Convolutional Neural Network [41], respectively. However, expensive labeled data is required to train these models. Although completely unsupervised methods

Table 2.4: Results on the SBHAR dataset in terms of accuracy.

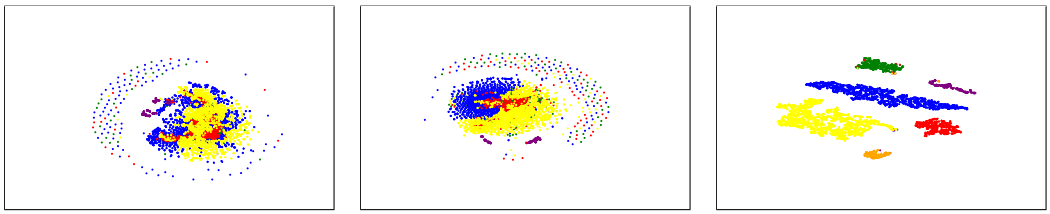
Method	**Accuracy(%)**
Probability SVM with Filter [15]	96.78
Convolutional Neural Network [41]	**98.7**
CNN Autoencoder with K-means [41]	55.0
PCA with K-means [41]	62.1
Proposed Method	92.68

do not need any labeled data, their performance are not comparable to supervised methods [41]. The proposed weakly supervised method is trained under limited su-



(a): Original Input Space (b): CNN Autoencoder (c): Proposed Method

Figure 2.3: t-SNE visualizations on the DG dataset: (a) Original input space, (b) Representation space from the CNN autoencoder and, (c) Representation space from the proposed method.



(a): Original Input Space (b): CNN Autoencoder (c): Proposed Method

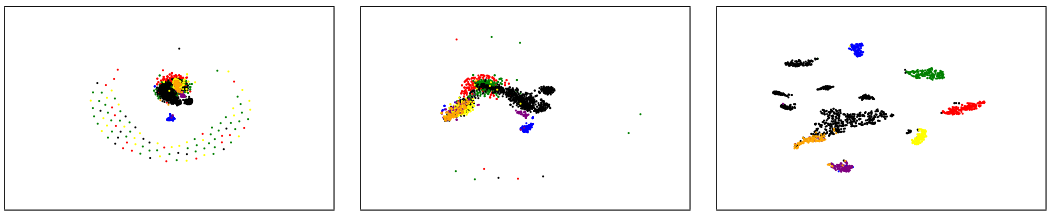
Figure 2.4: t-SNE visualizations on the WISDM dataset: (a) Original input space, (b) Representation space from the CNN autoencoder and, (c) Representation space from the proposed method.

pervision by using only the information about the similarity between the activities and achieves comparable results to supervised methods on the WISDM and SBHAR datasets while significantly outperforming the unsupervised methods.

2.4.4 Visualization of the Representation Space

In order to better understand the distribution of the activity vectors in the representation space, we applied t-sne [42] to map the representation vectors to two dimensions for visualization. The results are shown in Figures 2.3, 2.4, and 2.5 for dataset DG, WISDM and SBHAR, respectively.

Fig. 2.3(a), 2.4(a), and 2.5(a) show the activity sequences in the original input space. In DG and WISDM all the activities are mixed together. In SBHAR only the activity *lying* is separable from other activities.



(a): Original Input Space (b): CNN Autoencoder (c): Proposed Method

Figure 2.5: t-SNE visualizations on the SBHAR dataset: (a) Original input space, (b) Representation space from the CNN autoencoder and, (c) Representation space from the proposed method.

Fig. 2.3(b), 2.4(b), and 2.5(b) show the results produced by a temporal convolutional autoencoder. The idea is to let the autoencoder compress the input activity sequence into a representation vector and reconstruct the sequence from this representation. It is expected that this compressed representation contains key information of the input sequence that is useful to reconstruct itself and to differentiate it from other activities. The results show that after being mapped into the representation space by the autoencoder, some representation vectors are almost separable. In SBHAR the vectors of *lying*, *sitting* and *standing* are separated from other activities. In WISDM the cluster of *sitting* is well formed. Thus, it is clear that the separation between static posture activities $\{lying, sitting, standing\}$ and dynamic activities $\{jogging, walking, ascending\ stairs, descending\ stairs\}$ is almost solved by the mapping of the autoencoder. It confirms that the autoencoder can capture useful information of the activities, especially the difference between static and dynamic activities. However, the mapping of the autoencoder is not powerful enough to further group each activity into its own cluster and it is not able to capture the difference between FoG and normal activities, so in the DG dataset, the FoGs are still mixed with normal activities.

Fig. 2.3(c), 2.4(c), and 2.5(c) are the results generated by the proposed method. In DG a long thin cluster is built. The possible reason is that most of the normal activities in the experiment are normal *walking*, and no significant difference exists between different *walking* sequences. FoG is an abnormal activity that shows up during normal *walking*, thus FoG also carries the characteristics of normal *walking*. Due to this reason, a long thin cluster can represent the process of normal *walking* that transfers to FoG gradually. As shown in Fig. 2.3(c), FoGs stay on one side of the cluster, while normal *walking* activities stay on the other side. In WISDM and SBHAR each kind of activity is grouped into its own cluster which is clearly separable. Note that in Fig. 2.5(c) the transitions and unknown activities are viewed as an independent special activity and grouped into 6 clusters. This result confirms our assumption in Subsection 2.3.3 that transitions between different kinds of activities are essentially different, and the unknown activities represent an infinite space of arbitrary activities. Although we treat all the transitions and unknown activities the same way, a long distance for different kinds of transitions and a small distance within the same kind of transition is still maintained.

2.5 Conclusions

In this paper, we proposed a weakly supervised method that can learn to segment and recognize human activities under limited supervision without using explicit labels. The proposed method uses a segmentation and a recognition module built around a common siamese network architecture consisting of CNN and LSTM layers to capture temporal relations efficiently. We verified its effectiveness on three HAR datasets and further analyzed the learned mapping function by visualizing the activity vectors in the representation space. The results show that the proposed model can map the activity sequences into a space where the distance metric indicates the

similarity of the activities. The learned distance metric can be applied with different clustering algorithms and achieve state-of-the-art or comparable performance to supervised methods.

CHAPTER 3

Weakly Supervised Multi-Task Representation Learning for Human Activity Analysis

3.1 Introduction

The increased availability of sensors embedded in the environment and worn on the body opens up a vast potential to improve many aspects of life and the workplace, including health management, assistive technology, and workplace safety and efficiency. To realize this potential, it is essential to address the arising need for improved human activity analysis from the sensor data stream. Many different methods have been developed to address this need. While current machine learning (ML) methods have achieved impressive results, most of these methods have limitations in the following perspectives: (i) The methods focus on one aspect of the data by either only recognizing the activity [3] or only identifying the person who performed the activity [7]; (ii) Most current person identification methods are based on iris, face, fingerprint or gait identification, in which specific input or activity is needed from the user side [10, 9]; (iii) Pure supervised training of the model is used which requires large amounts of labeled data [21], that is often hard to come by, especially in personalized applications.

This paper presents an approach that attempts to mitigate these limitations. Intuitively, different persons perform activities in different ways. Distinct personal characteristics are commonly present in all the activities performed by the person. Thus, identifying a person with different types of activities can generalize the model to activity-based identification which does not need a specific user cooperation. More-

over, from the multi-tasking perspective, sharing knowledge between related tasks leads to learning generalized representations, helping to reduce the risk of overfitting one specific task. Since human activity recognition (HAR) and person identification are closely related, combining them into one multi-task model is reasonable and can be beneficial for both tasks. Furthermore, while learning of a selective representation is commonly achieved by supervised training, previous works in computer vision [43, 25] have shown the possibility to achieve similar performance in a weakly supervised manner while dramatically reducing the effort needed to obtain training data.

Based on these observations, this paper proposes a unified deep learning architecture centered around siamese networks and temporal convolutions for simultaneous HAR and activity-based person identification, which is trained using only the information about the similarity of the activities and the persons without knowing the explicit labels. In addition, we further expand the model to learn representations for additional attributes in the data along with HAR and person identification. The experiments show that our model is not restricted to any specific task, can be easily expanded to other related tasks, and achieves competitive performance compared to state-of-the-art methods , including fully supervised methods that take advantage of additional, explicit labels, on several datasets.

3.2 Related Work

3.2.1 HAR and Person Identification

There are two main directions in wearable sensor-based HAR, either handcrafted feature based methods or deep neural network (DNN) based methods. Handcrafted features are designed with domain knowledge. For example, [36, 15] used

statistical features, e.g. mean, variance and entropy, in their models. Features extracted from a wavelet transform were utilized in [19]. He and Jin [20] used features extracted by applying discrete cosine transform. The advantage of these features is that they can be derived from the signal easily and have been shown to be effective in the HAR system. However, domain knowledge is required to design the features manually. Recently, many HAR models adopt DNN to allow automatic feature extraction. Morales and Roggen [21] proposed a model consisting of convolutional neural network (CNN) and long short-term memory (LSTM) components. CNN is used here to capture local temporal relations while the memory states of LSTM ease the learning of long time scale dependencies. In [40], a hybrid approach was proposed, which used a deep belief network as an emission matrix of a hidden Markov model to model the sequence of human activities. These methods can automatically extract features from the data without any domain knowledge. But, they still require explicit labels to supervise the training of the model.

Many person identification methods are based on iris, face, and fingerprint. Those kinds of methods require specific cooperation or explicit action/input from the user side, e.g. standing in front of a camera, looking at a specific point, etc. [44], while gait recognition-based person identification, e.g. [9, 45, 8], enables an inexpensive, convenient, and unobtrusive way to complete the task. However, gait-based methods assume that walking is the only activity to be performed during the identification. In many real-world applications, this assumption may not hold. There have been only a few studies that are identifying the person based on various activities recorded by sensors. Kwapisz, Weiss, and Moore [7] proposed a model which used handcrafted features and decision trees. Their model addressed the biometric identification task by analyzing four types of dynamic activities (walking, jogging, ascending and descending stairs). Elkader et al [44] expanded the person identification method from a limited

number of specific activities to a set of various normal daily activities. However, these works still focus on only one aspect of the data.

The most closely related works are perhaps [46, 47]. In [46], Reddy et al. proposed a method to first identify the person's states: standing, sitting, or walking, then separate SVM based models are used for identifying the person for each of the three mentioned states. Hernandez, McDuff, and Picard [47] collected the experimental data from a wrist worn smartwatch, and their proposed model identifies the person and three static body postures (sitting, standing, and lying) at the same time. These two works address two tasks, but their methods are based on a very small number of simple activities.

3.2.2 Siamese Networks and Temporal Convolution Networks

We take inspiration from siamese architectures, and temporal convolutional networks (TCN) to design our model that can efficiently capture the temporal patterns and compute the semantic similarity between the pairs of data sequences.

A siamese network [23] is a neural network with dual branches and shared weights. As illustrated in Fig. 3.1, it processes an input pair $\{x_a, x_b\}$ and yields a pair of comparable representation vectors $\{H_a, H_b\}$. The distance between the comparable representation vectors is then used as the semantic similarity of the input pair. The siamese architecture is widely used in many domains. Originally [23], a siamese network was used to verify signatures. Mueller and Thyagarajan [24] used a siamese recurrent neural networks to measure the semantic similarity between a pair of sentences. In [25], a siamese CNN is proposed to learn a complex similarity metric for face verification. In other areas, the siamese architecture has been applied in unsupervised acoustic model learning [26, 48, 49], image recognition [50] and object tracking [43].

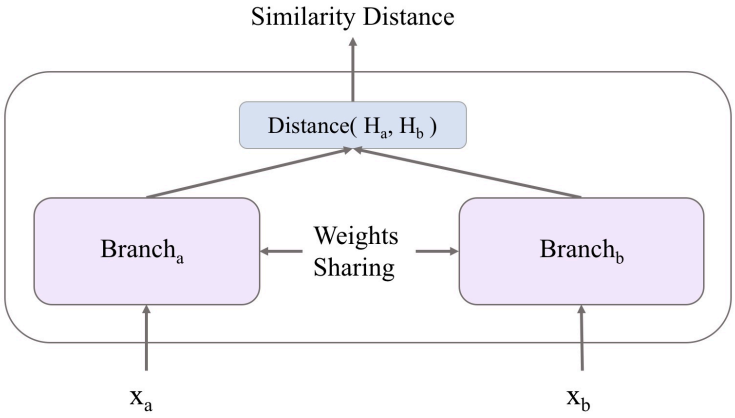


Figure 3.1: The basic architecture of a siamese network.

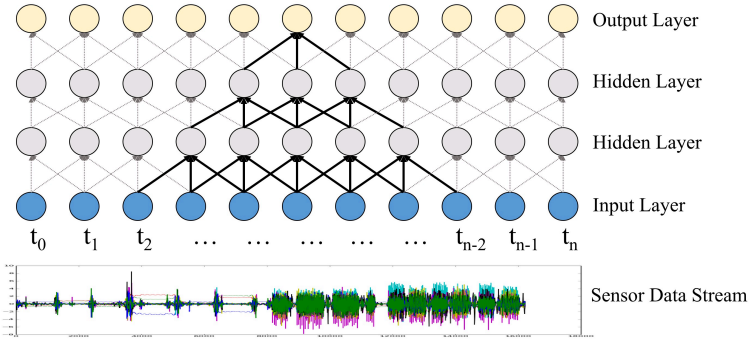


Figure 3.2: The basic architecture of a temporal convolutional network (TCN).

As shown in Fig. 3.2, a TCN uses a hierarchy of convolutions to abstract the temporal relations of the data stream at different time scales. It has been successfully applied in many different areas, e.g., computer vision (video data stream), natural language processing (speech or text data), etc. In [51], Oord et al. proposed a model that can generate raw speech signals. In [52, 53], TCN has been used to segment and detect action in the video. In [54], a character-level TCN is used to classify text.

3.3 Proposed Method

Our work differs from previous works in several important ways. First, unlike most of the previous methods, our proposed model can identify the activity and the

person simultaneously and can be systematically expanded to identify additional types of attributes. Second, in comparison to most of the previous wearable sensor-based person identification methods, which have been customized to either only focused on gait information, or only used a limited number of dynamic activities or static body postures, our model uses a task-agnostic structure and is here evaluated on four diverse public datasets [55, 56, 15, 18]. These datasets, respectively, contain 12, 12, 7, or 6 types of different activities performed by 9, 10, 30, or 36 different persons. This evaluation offers a more realistic and challenging scenario, and through the use of a wider range of activities may lead to a more robust identification model. Moreover, since in the daily routine of a person in real life many different activities will be performed, an identification system that is restricted to a very small number of activities might fail to work, and even in situations where it can still make correct classifications can only take advantage of a small part of the available data. Finally, our model, by taking advantage of the siamese architecture, can be trained in a weakly supervised manner, hence no explicit labels are needed.

To achieve these capabilities, we frame the problem as learning an invariant mapping that maps the input data sequence into a semantic representation space. The learning process relies only on the relationship of the data sequences in the input pair, therefore the learned model will map two data sequences either to the same area in the representation space if the two data sequences are semantically similar, or to different areas if the data sequences are semantically dissimilar. Formally, given a pair of sensor data sequences $\{x_a, x_b\}$, the aim is to learn a mapping f that maps the pair $\{x_a, x_b\}$ into a representation space such that:

$$H_{x_a} = f(x_a) \tag{3.1}$$

$$H_{x_b} = f(x_b) \tag{3.2}$$

The distance between the representation pair $\{H_{x_a}, H_{x_b}\}$ approximates the semantic similarity of the input pair $\{x_a, x_b\}$. Specifically, our proposed model learns two such mappings:

$$f^{act} : \begin{cases} x_a \rightarrow H_{x_a}^{act} \\ x_b \rightarrow H_{x_b}^{act} \end{cases} \tag{3.3}$$

$$f^{pers} : \begin{cases} x_a \rightarrow H_{x_a}^{pers} \\ x_b \rightarrow H_{x_b}^{pers} \end{cases} \tag{3.4}$$

The mapping f^{act} is based on the activity similarity of x_a and x_b, such that if x_a and x_b belong to the same type of activity, then $H_{x_a}^{act}$ and $H_{x_b}^{act}$ will stay together, while $H_{x_a}^{act}$ and $H_{x_b}^{act}$ will stay away from each other, if x_a and x_b are from different types of activities. The mapping f^{pers} is based on the performer of the activity, such that if x_a and x_b are performed by the same person, then $H_{x_a}^{pers}$ and $H_{x_b}^{pers}$ will stay together, while $H_{x_a}^{pers}$ and $H_{x_b}^{pers}$ will stay away from each other if x_a and x_b are performed by different persons. The mappings f^{act} and f^{pers} preserve different semantic relationships between the input data sequences. The proposed model learns these two mappings at the same time.

3.3.1 TCN Blocks

Suppose, we are given a pair of sensor data sequences $\{x_a, x_b\}$: $x_a = (\mathrm{x}_{a_1}, ..., \mathrm{x}_{a_{\tau'}})$ and $x_b = (\mathrm{x}_{b_1}, ..., \mathrm{x}_{b_{\tau''}})$, where τ' and τ'' denote the time length of the signals and $\mathrm{x}_{a_t} = [s_t^1, ..., s_t^n]$ is the n-dimensional sensor reading in sequence x_a at time t. The learned mappings are then defined as follows:

$$f^{act} : \begin{cases} (\mathrm{x}_{a_1}, ..., \mathrm{x}_{a_{\tau'}}) \rightarrow H_{x_a}^{act} \\ (\mathrm{x}_{b_1}, ..., \mathrm{x}_{b_{\tau''}}) \rightarrow H_{x_b}^{act} \end{cases} \tag{3.5}$$

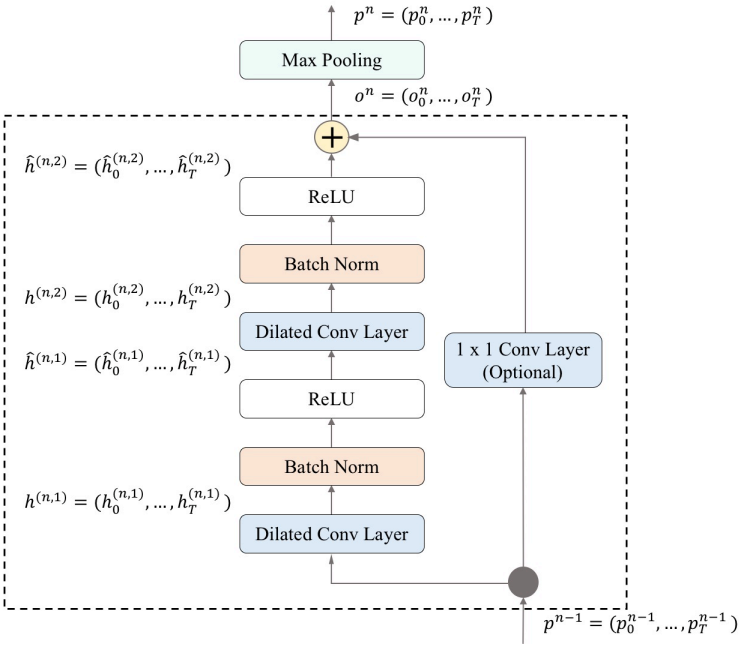


Figure 3.3: A TCN block with two convolutional layers.

$$f^{pers} : \begin{cases} (x_{a_1}, \ldots, x_{a_{\tau'}}) \rightarrow H_{x_a}^{pers} \\ (x_{b_1}, \ldots, x_{b_{\tau''}}) \rightarrow H_{x_b}^{pers} \end{cases} \tag{3.6}$$

The TCN architecture is adopted as the basic building block for abstracting the sequence due to its ability to efficiently abstract time series data at different time scales. As illustrated in Fig. 3.3, each TCN block is composed of a series of transformations, which includes the dilated temporal convolutions with dilation rate d, batch normalization, non-linearity $g(\cdot)$, and residual connection $\oplus$. Assume that the proposed model contains a sequence of N TCN blocks where each block contains L convolutional layers with m different feature maps of width k, and the parameters of the feature maps are w. For the n^{th} block, the input p^{n-1} is the max-pooled output from the $(n-1)^{\text{th}}$ block, and the temporal convolutions that we apply in each TCN block to capture the patterns over the course of an activity are constructed from the following components.

3.3.1.1 Dilated Convolutional Layer

The advantage of dilated convolutions is that they support faster expanding receptive fields without losing resolution or coverage [28]. Batch Normalization (BN) is applied after each dilated convolutional layer and before the non-linear activation function. BN can accelerate the learning process by reparameterizing the underlying optimization problem to make it more stable and smooth [57]. Formally, in the l^{th} convolutional layer of the TCN block, the computation is then defined as follows:

$$h_t^{(n,l)} = \begin{cases} \sum_{i=-\frac{k-1}{2}}^{\frac{k-1}{2}} w_i \cdot p_{t-d\cdot i}^{n-1}, & \text{if } l = 1. \\ \sum_{i=-\frac{k-1}{2}}^{\frac{k-1}{2}} w_i \cdot \hat{h}_{t-d\cdot i}^{(n,l-1)}, & \text{otherwise.} \end{cases} \tag{3.7}$$

$$\hat{h}_t^{(n,l)} = g(h_t^{(n,l)}) \tag{3.8}$$

where $h^{(n,l)}$ is the output of the convolutions and $\hat{h}^{(n,l)}$ is the output of the non-linear activation function.

3.3.1.2 Residual Connections

The output of the TCN block, o^n, is the sum of the result of the last convolution $\hat{h}_t^{(n,L)}$ and the input of the block, p^{n-1}. However, in TCN blocks, the shapes of the input tensor and the output tensor can be different [58]. To address this problem, if $\hat{h}_t^{(n,L)}$ and p_t^{n-1} have different shapes, a 1x1 convolution will be used as the residual connection. Otherwise, an identity function will be used as the residual connection:

$$o_t^n = \hat{h}_t^{(n,L)} \oplus p_t^{n-1} \tag{3.9}$$

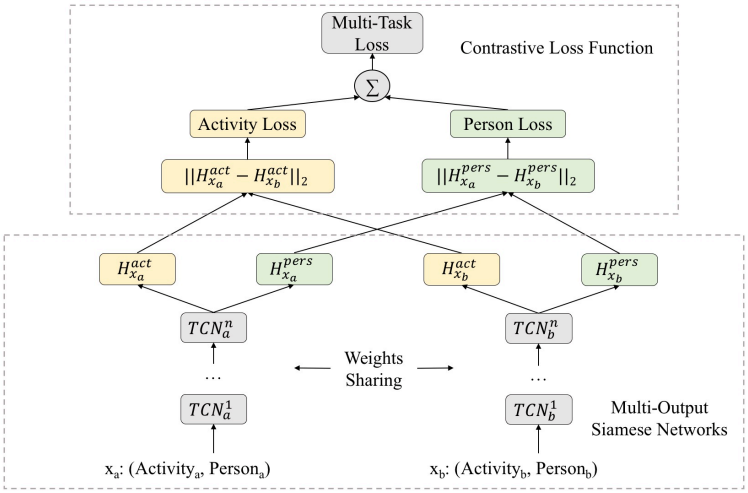


Figure 3.4: A dual-output siamese networks.

The temporal max pooling with width 2 is used between two consecutive TCN blocks to reduce the size of the time dimensions while introducing slight translational invariance in time:

$$p_t^n = max(o_{t-1}^n, o_t^n) \tag{3.10}$$

3.3.2 Multi-Output Siamese Networks

The architecture of the proposed model is outlined in Fig. 3.4. It contains two networks TCN_a and TCN_b. Each network has n TCN blocks, shares the weights, and processes one of the data sequences in the input pair $\{x_a, x_b\}$. In order to disentangle the activity and person information extracted by the TCN networks, two fully-connected (FC) layers are connected to the TCN networks and are responsible to process the activity representations $\{H^{act}_{x_a}, H^{act}_{x_b}\}$ and person representations $\{H^{pers}_{x_a}, H^{pers}_{x_b}\}$, respectively. The weights of the FC layers are shared within the same representation learning task.

This model is trained using 4-tuples $\{x_a, x_b, y^{act}, y^{pers}\}$, where x_a and x_b are the data sequences of the input pair. $y^{act}, y^{pers} \in \{0, 1\}$ denote the semantic relationships

of the input pair, where $y^{act} = 0$ or $y^{pers} = 0$ denotes that $\{x_a, x_b\}$ is a semantically negative pair, i.e. they are maximally dissimilar in terms of the corresponding property. If $y^{act} = 0$, x_a and x_b are different kinds of activities; if $y^{pers} = 0$, x_a and x_b are performed by different persons. $y^{act} = 1$ or $y^{pers} = 1$ denotes that $\{x_a, x_b\}$ is a semantically positive pair. If x_a and x_b are the same kind of activity, $y^{act} = 1$; if x_a and x_b are performed by the same person, $y^{pers} = 1$.

3.3.3 Contrastive Loss Function

Given the input pair $\{x_a, x_b\}$, the model outputs a pair of activity representations, $\{H^{act}_{x_a}, H^{act}_{x_b}\}$, and a pair of person representations $\{H^{pers}_{x_a}, H^{pers}_{x_b}\}$. The loss function used to train the model is also defined on the pairs. The similarity distance $Dist_s$ between the input pair is measured by the euclidean distance between their representation pair:

$$Dist_s(x_a, x_b) = ||H_{x_a} - H_{x_b}||_2 \tag{3.11}$$

To make the notation clearer, $Dist_s(x_a, x_b)$ is rewritten as D. Then the loss function for each of the categorizations used for training is defined as:

$$L(x_a, x_b, y) = \sum_{i=1}^{N} L^i(x^i_a, x^i_b, y^i) \tag{3.12}$$

$$L^i(x^i_a, x^i_b, y^i) = y^i L_s(x^i_a, x^i_b) + (1 - y^i) L_d(x^i_a, x^i_b) \tag{3.13}$$

where (x^i_a, x^i_b, y^i) is the i-th sample in the data set. L_s, L_d are the loss terms for the positive pair ($y = 1$) and the negative pair ($y = 0$). The forms of L_s, L_d are given by:

$$L_s = \frac{1}{2}(D)^2 \tag{3.14}$$

$$L_d = \frac{1}{2}\{max(0, \delta - D)\}^2 \tag{3.15}$$

where δ is a margin hyperparameter. It defines that the negative pairs contribute to the loss function if their distance is smaller than the margin δ. Then, the loss function

is applied in each representation space, and the weighted sum of the loss function in each representation space is defined for the final multi-task model (α and β are the corresponding weights for each task):

$$\begin{aligned} &\mathcal{L}(x_a, x_b, y^{act}, y^{pers}) = \\ &\alpha \cdot L^{act}(x_a, x_b, y^{act}) + \beta \cdot L^{pers}(x_a, x_b, y^{pers}) \end{aligned} \tag{3.16}$$

To train the network, we use the standard backpropagation algorithm with stochastic gradient descent. All the parameters are initialized to small values. The start learning rate is $lr = 0.05$ and an exponential decay function is applied to the learning rate every 10000 steps with a decay rate of 0.95. The network is tested on validation data after each epoch. If the validation error stopped decreasing for a predefined number of epoch, training is finished.

3.3.4 Cluster Construction

After all the parameters are learned, we can use the trained model to provide metrics for a wide range of different clustering algorithms. Because the trained model can map the data sample x into the representation space, where the representation vector H is positioned such that data with the same semantic meaning are located close to each other, clusters in this space should capture the corresponding property. More specifically, in our experiments, the trained model maps the data samples into the activity representation space and the person representation space. In the activity representation space, the data samples will be grouped into clusters according to the activity type. In the person representation space, the data samples will be grouped into clusters according to the identity of the person. Hence, after mapping the data samples to the more clustering-friendly representations, different clustering algorithms can be used on these learned representations. In our experiments, K-means is employed as the clustering method.

3.4 Evaluation and Experiments

In order to evaluate the effectiveness of the proposed model, we use four public datasets that contain raw sensor data sequences of different human activities performed by different persons. We conduct activity clustering and person clustering on the learned representations as described in Section 3.3.4.

3.4.1 Datasets

The datasets used here are selected from widely used benchmark datasets [59] as the ones that contain a good number of different persons performing numerous diverse activities. Those datasets are recorded by various sensors, e.g. accelerometer, gyroscope, magnetometer etc, and include human activities in different scenarios. All the sensor data sequences are segmented with a sliding window as described with each of the datasets below.

The **PAMAP2** dataset is collected from 9 participants performing 12 activities over a total of 10 hours. It includes sport exercises (rope jumping, nordic walking etc), and household activities (vacuum cleaning, ironing etc). One heart-rate monitor and three inertial measurement units (IMUs) located on the chest, dominant wrist and ankle were used to record the heart rate, accelerometer, gyroscope, magnetometer, and temperature data. We replicate previous work [37, 4] to downsample the data from 100Hz to 33.3Hz, and use a sliding window of 5.12 seconds with one second step size.

The **MHEALTH** dataset contains data recorded from 10 volunteers carrying out 12 physical activities, including primitive body parts movements (waist bends forward, frontal elevation of arms etc), and composite body movements (cycling, jumping front and back etc). The data is collected by using three sensors placed on the subject's chest, right wrist and left ankle to record accelerometer, gyroscope,

and magnetometer signals. The chest sensor also records 2-lead ECG signals. The sampling rate of all sensing modalities is 50Hz. As in previous work [60], we use a sliding window of 5 seconds with a step size of 2.5 seconds.

The **SBHAR** dataset provides data gathered from 30 participants performing 6 basic activities, such as walking and lying, and 6 postural transitions, such as stand-to-sit, sit-to-lie. In our experiment, we consider all the postural transitions as one general transition. The data was collected by placing a smartphone on the waist of the participant, and the inertial sensors in the smartphone were used to record the accelerometer and gyroscope data at a sampling rate of 50Hz. As used in the previous work [15], we use a sliding window of 2.56 seconds with a step size of 1.28 seconds.

The **WISDM** dataset contains data collected with one accelerometer in a smartphone from 36 volunteers carrying out 6 activities, including jogging, climbing stairs, etc. The sampling rate is 20Hz. We use the same settings as used in [18] to set the sliding window size to be 10 seconds without overlap.

3.4.2 Performance Metrics

To compare with previous works, we use mean F1-score (F_m) and clustering accuracy as the metrics. F_m is defined as follows:

$$F_m = \frac{2}{|C|} \sum_{i=1}^{C} \frac{Precision_i \cdot Recall_i}{Precision_i + Recall_i} \tag{3.17}$$

where $i = 1, \ldots, C$ is the set of classes. For the given class i, $Precision_i = \frac{TP_i}{TP_i + FP_i}$, $Recall_i = \frac{TP_i}{TP_i + FN_i}$; TP_i and FP_i denote the number of true positives and false positives, and FN_i is the number of false negatives.

Table 3.1: Results on PAMAP2 in terms of F_m

Methods	**Activity**	**Person**
LSTM-F [4]	0.9290	-
CNN [4]	0.9370	-
DNN [4]	0.9040	-
PCA + k-Means	0.4244	0.2040
3-NN	-	0.9416
5-NN	-	0.9014
Decision Tree	-	0.9781
Proposed Multi-Task Method	**0.9893**	**0.9881**

Table 3.2: Results on MHEALTH in terms of Accuracy

Methods	**Activity**	**Person**
CNN-1D [61]	0.9809	-
CNN-2D [61]	0.9829	-
CNN-pff [62]	0.9194	-
PCA + k-Means	0.4850	0.2361
3-NN	-	0.8540
5-NN	-	0.8380
Decision Tree	-	0.6714
Proposed Multi-Task Method	**0.9957**	**0.9948**

3.4.3 Results

We used the same model architecture with 3 TCN blocks across all the experiments. A shorthand description of the shared layers is: TCN (128) $-P-$ TCN (128) $-P-$ TCN (128), where TCN(128) denotes a TCN block with 128 feature maps, and P a max-pooling layer. The internal structure of a TCN block is the same as

Table 3.3: Results on SBHAR in terms of Accuracy

Methods	**Activity**	**Person**
Probability SVM [15]	0.9580	-
Probability SVM with Filter [15]	0.9678	-
CNN [41]	0.9870	-
PCA + k-Means	0.5282	0.1299
3-NN	-	0.5330
5-NN	-	0.5123
Decision Tree	-	0.4957
Proposed Multi-Task Method	**0.9885**	**0.8892**

Table 3.4: Results on WISDM in terms of Accuracy

Methods	Activity	Person
CNN with partial weight sharing [3]	0.9688	-
DBN+HMM [40]	**0.9823**	-
CNN+stat. features [6]	0.9332	-
Neural Net* [7]	-	0.6950
Decision Tree* [7]	-	0.7220
PCA + k-Means	0.5181	0.2430
3-NN	-	0.2651
5-NN	-	0.2470
Decision Tree	-	0.2189
Proposed Multi-Task Method	0.9576	**0.8112**

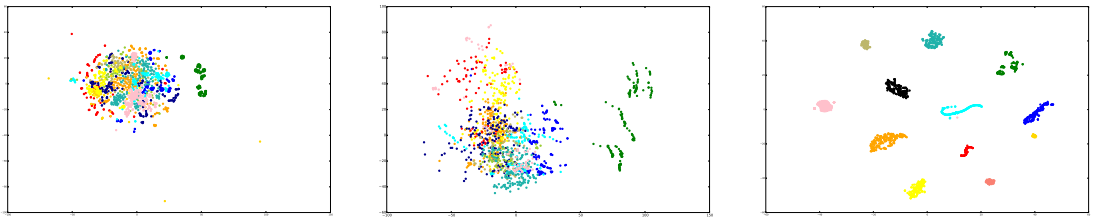
* Experiment results based on 4 activities.

illustrated in Figure 3.3. In addition, above the last TCN block, one FC layer with 256 hidden nodes is used for each single representation learning task.

The experimental results are summarized in Tables 3.1, 3.2, 3.3, and 3.4. In addition to previous published works which use fully supervised learning, we also used principal components analysis (PCA), k-nearest neighbors ($k = 3, 5$), and a decision tree algorithm (C4.5) as baselines. As shown in the result tables, our proposed multi-task method achieved competitive performance on both tasks compared with the supervised single-task approaches in most situations. Since most previous methods were applied only to one of the tasks, only the performance results for that task are listed in the tables.

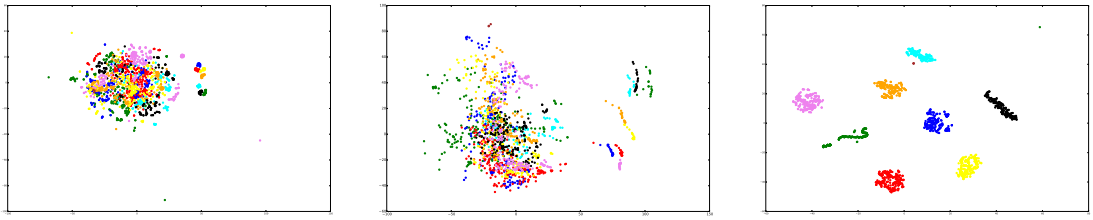
3.4.4 Visualization and Analysis

To further analyze the representations learned by the proposed multi-task method and compare it with other embedding techniques, we used t-sne [42] to visualize the activity and person representation spaces. Due to the space limitation, we only list the visualizations of the proposed multi-task method and PCA on two representative datasets: PAMAP2 and SBHAR. PAMAP2 contains complex activities, which makes



(a): Original input space (b): PCA (c): Proposed method

Figure 3.5: Visualizations on the activity aspect of PAMAP2.

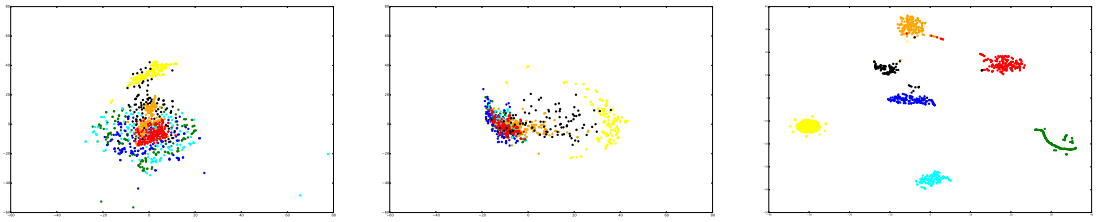


(a): Original input space (b): PCA (c): Proposed method

Figure 3.6: Visualizations on the person aspect of PAMAP2.

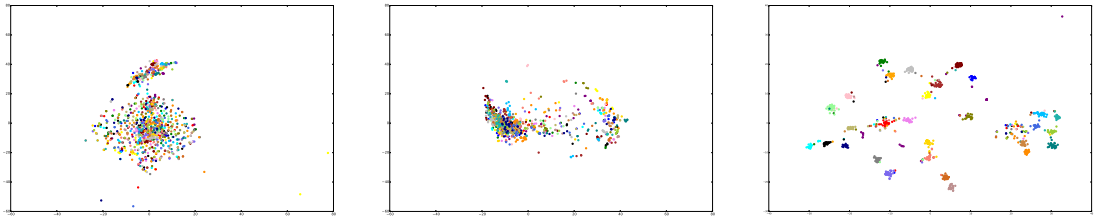
it difficult to model. SBHAR contains unbalanced class numbers in the two aspects of the data. In particular, it includes 30 persons who perform 7 activities, which leads to unbalanced sample sets across the two tasks in that there are significantly fewer data samples for the person classes than for the activity classes. This, in turn, leads to bias and difficulties learning these two highly unbalanced tasks simultaneously.

In Fig. 3.5, 3.6, 3.7 and 3.8, different colors denote different true semantic labels in the datasets. From these figures, we can reach the following conclusions: (i) The proposed multi-task method can effectively disentangle different semantic representations from the data. (ii) The learned representation clusters are compact and clearly separated in each representation space. Thus, downstream tasks like clustering can benefit from it and achieve promising performance. (iii) For those datasets that con-



(a): Original input space (b): PCA (c): Proposed method

Figure 3.7: Visualizations on the activity aspect of SBHAR.

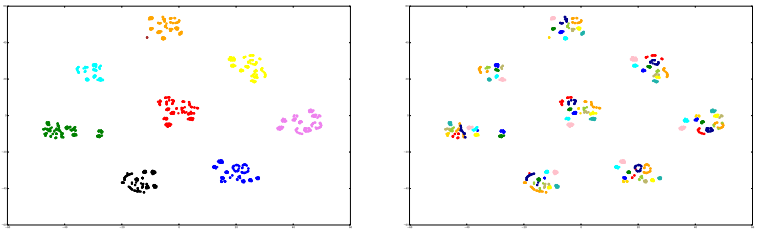


(a): Original input space (b): PCA (c): Proposed method

Figure 3.8: Visualizations on the person aspect of SBHAR.

tain unbalanced class numbers in different aspects of the data, the qualities of the learned representations are also unbalanced. For example, in Fig. 3.7 and 3.8, the activity representations are more compact within the same cluster, and different clusters are clearly separated with each other. But in the person representation space, some data samples stay away from their corresponding clusters, and some clusters stay relatively close to each other. The imbalance is also reflected on the performances. As shown in Table 3.3, the accuracy on the activity recognition task is 10% higher than on the person identification task.

After evaluating the effectiveness of the learned representations for each task, we also visualize the output from TCN, which we consider a more general representation because it is learned by the shared layers in the model without aiming at any specific task. As shown in Fig. 3.9(a), 3.9(b), the general representations form a two-level



(a): General rep. colored by persons

(b): General rep. colored by activities

Figure 3.9: Visualizations on the general representations of PAMAP2.

structure. At the higher level, illustrated in Fig. 3.9(a), the clusters are grouped in accordance with the identity of the persons. At the lower level, shown in Fig. 3.9(b), in each person's cluster, different activities are further grouped into different smaller sub-clusters.

3.4.5 Ablation Studies

As the experimental results in Section **??** show, the proposed multi-task model can learn the two related tasks (activity recognition and person identification) successfully. To further understand the effect of the multi-task training framework, a series of ablation experiments are performed to separately measure the influence of multi-task learning. In the ablation experiments, the original objective of the multi-task learning is transformed into two separate objectives and two separate single-task learning models are built based on the same underlying siamese network architecture. Thus, each single-task model is trained to learn only one task, namely activity recognition (AR) or person identification (PI). Comparing these to the model trained using multi-task data should provide insight into the benefit of multi-task training and thus any cross-fertilization occurring between the two tasks. The experiment results of the ablation studies are provided in Table 4.5.

Table 3.5: Ablation studies on effect of multi-task learning.

Methods	Activity	Person
PAMAP2		
Single-Task Model AR	0.9856	–
Single-Task Model PI	–	0.9812
Proposed Multi-Task Method	**0.9893**	**0.9881**
MHEALTH		
Single-Task Model AR	0.9756	–
Single-Task Model PI	–	0.9889
Proposed Multi-Task Method	**0.9957**	**0.9948**
SBHAR		
Single-Task Model AR	0.9436	–
Single-Task Model PI	–	0.8138
Proposed Multi-Task Method	**0.9885**	**0.8892**
WISDM		
Single-Task Model AR	**0.9629**	–
Single-Task Model PI	–	0.8080
Proposed Multi-Task Method	0.9576	**0.8112**

As shown in Table 4.5, expanding the model from single-task learning to multi-task learning does not cause performance loss and degradation in most cases. On the contrary, the performance of multi-task training is most of the time better than the one of the single-task models. The only exception here is in the case of the WISDM dataset where a small drop in accuracy for activity recognition goes along with an increase in accuracy for person identification. One possible reason is that the WISDM dataset is collected with one single accelerometer with a relatively low sampling rate, providing significantly less data per time unit compard to the other datasets. Due to this, the time window used in this dataset is also significantly longer (10 s), leading to a much coarser grained set of labeled data points, making it potentially harder to arrange them in spaces that clearly separate them based on multiple semantic criteria due to the larger amount of overlap in the data. As a result, it might be the case here that the data does not contain sufficient information for learning the more complex multi-task model, necessitating the observed trade-off between activity

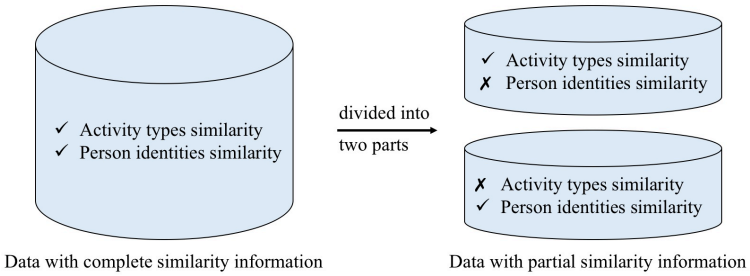


Figure 3.10: Generation of data with partial similarity information from the original data for partial information experiments.

recognition and person identification. However, it is important to note here that the drop in activity recognition performance is relatively limited and smaller than the obtained gain in person identification accuracy. This result confirms the assumption that activity recognition and person identification are closely related tasks, thus, sharing knowledge and learning a generalized representation between them can be beneficial for both tasks.

3.4.6 Multi-Task Learning with Partial Similarity Information

Having analyzed the effect of the multi-task framework, we now consider another, more challenging scenario. Previous experiments are all based on the assumption that the similarity information of the activity types and the person identities is fully visible to the model. However, in many real world applications it will be impossible for the model to have complete and perfect information about the similarities between all data items and for all attributes of interest. Therefore, in this section, another additional constraint is imposed on the proposed multi-task method. We conducted experiments where the model is trained with only partial observations of the similarity information about the data. To be more specific, as illustrated in Fig. 3.10, we divided the dataset into two parts of equal size. One part of the data only provides the similarity information of the activity types, while the other part

of the data only provides the similarity information of the person identities. It is important to note here that this implies that no single data item in the training set contained similarity information for both tasks and this can thus be seen as similar to a case where data was separately collected for the two tasks and no re-assessment of the similarity for the other task was performed after collection. Then, the proposed multi-task method is trained to learn both AR and PI tasks with this data. The experimental results are summarized in Table 3.6.

Table 3.6: Multi-task learning with partial similarity information.

Methods	**Activity**	**Person**
PAMAP2		
Single-Task Model AR	0.9856	–
Single-Task Model PI	–	0.9812
Proposed Multi-Task Method with Complete Similarity Information	**0.9893**	**0.9881**
Proposed Multi-Task Method with Partial Similarity Information	0.9849	0.9879
MHEALTH		
Single-Task Model AR	0.9756	–
Single-Task Model PI	–	0.9889
Proposed Multi-Task Method with Complete Similarity Information	**0.9957**	**0.9948**
Proposed Multi-Task Method with Partial Similarity Information	0.9744	0.9643
SBHAR		
Single-Task Model AR	0.9436	–
Single-Task Model PI	–	0.8138
Proposed Multi-Task Method with Complete Similarity Information	**0.9885**	**0.8892**
Proposed Multi-Task Method with Partial Similarity Information	0.9618	0.8247
WISDM		
Single-Task Model AR	**0.9629**	–
Single-Task Model PI	–	0.8080
Proposed Multi-Task Method with Complete Similarity Information	0.9576	**0.8112**
Proposed Multi-Task Method with Partial Similarity Information	0.9509	0.7791

As shown in Table 3.6, for both tasks, despite the data only containing partial similarity information for the multi-task model and thus only providing information regarding one task from each sample, the performance of the multi-task method is relatively close to the performance of the multi-task method using the complete sim-

ilarity information for each data item and outperforms a few single-task learners trained on the data with full information from the ablation study. This indicates that our proposed method, which is trained to learn both tasks with partial similarity information is capable of effectively utilizing cross-task information to achieve performance that is competitive with other models trained with complete similarity information for all tasks.

It is worth to mention that one of the reasons for the model performance loss on SBHAR and WISDM is that these two datasets contain unbalanced class numbers, SBHAR contains 30 persons performing 7 activities and WISDM contains 36 persons performing 6 activities. This leads to the situation where each person class has significantly fewer data samples than an activity class. Moreover, when the datasets are divided into two parts of equal size, each part only contains partial information of the data, and this further reduces the size of the data samples that can be used for the PI task learning. Therefore, without enough training data, the performance of the model will be degraded.

3.4.7 Attribute Representation Learning

In order to evaluate the scalability and robustness of the method, we also expanded the proposed multi-task model to include additional attribute representation learning. We selected PAMAP2 to evaluate the expanded model, because this dataset provides extra attributes of the data that the model can learn. Here, we choose the gender of the person attribute as another representation learning task. The expanded model is illustrated in Fig. 3.11 and shows the simplicity of expanding the proposed network to include additional representation and similarity learning tasks.

The experiment results are summarized in Table 3.7. As we can see from the table, the expanded model works well on all three tasks. Three types of different

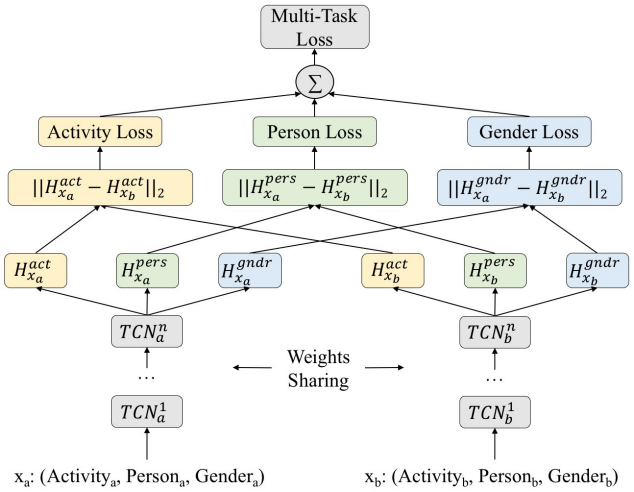


Figure 3.11: A tri-output siamese network.

Table 3.7: Results of proposed multi-task method on PAMAP2 with attribute representation learning in terms of F_m

Methods	**Activity**	**Person**	**Gender**
Single-Task Model AR	0.9856	–	-
Single-Task Model PI	–	0.9812	-
Single-Task Model Gender	–	–	**0.8678**
Proposed Multi-Task Method with Dual-Output	**0.9893**	**0.9881**	-
Proposed Multi-Task Method with Tri-Output	0.9798	0.9827	0.8637

semantic representations are learned successfully. However, the unbalanced performance is also presented in this model, i.e. the performance on gender attribute learning is around 11% lower than the performances on the other two tasks.

3.5 Conclusions and Future Work

In this paper, we propose a weakly supervised multi-output representation learning approach that is built around a siamese architecture consisting of temporal convolutions to capture multiple semantic similarities in the data simultaneously. The clustering results on the learned representations achieved promising performance and even outperformed supervised models in most situations. The visualization analysis

demonstrates that the proposed approach succeeds in disentangling the semantic representations, while preserving the similarity metrics in all the representation spaces. Moreover, a series of ablation studies analyzed the effect of the multi-task framework and showed that the use of multi-task training in this architecture most of the time improves performance over single task training, thus illustrating the frameworks ability to efficiently share information between the tasks. Additional experiments showed that the proposed method can also be applied to data with only partial similarity information, can be expanded to learn new, additional task easily, and achieve competitive results in both cases. The comprehensive experiments give useful insights for the proposed multi-task method. Future work will be focused on designing a method that learns to adapt to the unbalancing aspects in the data automatically.

CHAPTER 4

Unsupervised Embedding Learning for Human Activity Recognition

4.1 Introduction

The typical process of sensor-based human activity recognition (HAR), as shown in Figure. 4.1, consists of three important stages: data segmentation, feature extraction, and recognizing the type of the activity. Extensive studies have been conducted in all the stages of the HAR process [16]. However, existing HAR methods rely heavily on labeled data to supervise the model training and to perform the recognition [40], thus a huge challenge for HAR system is collecting annotated data. In the meantime, more and more wearable devices, smartphones, smart watches, etc., are used in people's daily lives. These wearable devices are usually equipped with various sensors, such as accelerometers, gyroscope, GPS sensors etc., which can provide a massive amount of unlabeled sensor activity data. Due to the above mentioned facts, most of the existing HAR systems can not take advantage of the accessible unlabeled data efficiently. Therefore, we aim at developing an unsupervised method to leverage the unlabelled data to recognize the physical activities without requiring the labels.

The Autoencoder (AE) [63] is an unsupervised learning framework to find efficient encodings of data. It encodes important features of the inputs into a hidden representation, and then reconstructs the inputs based on their hidden representations. It is applied for dimension reduction, deep hierarchical model pre-training etc. Because of its simplicity and efficiency, we also design our model based on the AE architecture. Yet only using reconstruction to guide the learning can lead the AE

to encode a significant amount of unnecessary information, such as task-irrelevant information, or even destructive information, such as noise.

Motivated by this observation, our approach utilizes the intrinsic properties of the sensor activity data to project the data into a clustering-friendly embedding space. Two fundamental observations contribute to the design and formation of this space.

Firstly, slow feature analysis [11] finds that many properties in the real world change slowly over time. Physical objects have inertia and their states usually change gradually and infrequently. This rule also applies to human activity. In most situations, the period of a person performing an activity will take a a relatively significant amount of time. It is rare that a person will switch between different activities very frequently. On the other hand, while during the course of an activity the type of the activity remains the same, the body pose of the activity varies over the time. Hence, we include the temporal coherence property in the loss function, which allows the model to learn the essential features of the activity and ignore the irrelevant temporal details in the body pose.

Secondly, distinct activity and person characteristics are commonly co-present in the sensor data. For the HAR task, only the activity characteristics are relevant. For example, two persons can walk in two different styles, but people can still identify they are performing the same type of activity: walking, because the activity-relevant characteristics are used in the recognition process and the irrelevant person characteristics are disregard. Based on this property, another local neighborhood based objective function, which aims at removing irrelevant personal or individual details in the data, is used to guide the learning of the model.

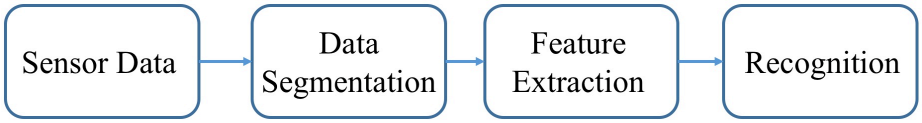


Figure 4.1: The typical process of HAR.

4.2 Related Work

Many works have been proposed to recognize human activity with wearable sensor data. As illustrated in Figure. 4.1, the first step in HAR is to segment the sensor data sequence. One common method used to segment the sequence is to use a sliding window [2] . We also adopted this method in our model for its simplicity and tractability. Feature extraction and recognition are conducted on the segmented data. To recognize the activity type, discriminative features are needed. They can be designed with domain knowledge or extracted automatically using, for example, neural networks (NN) [64].

Handcrafted features, when designed properly, have proven to be very useful in HAR systems. In [18], statistical features are derived from the time series sensor data. In [20] and [19], discrete cosine transform and wavelet transform are used to convert the sensor signal from the time domain into the transformed domain. Then features derived from the transformed domains are applied in the recognition process.

With the development of more competent deep learning techniques, and in particular NNs, automatic feature extraction has become another effective way to obtain discriminative features. In [40], a deep belief network was used as the emission matrix of a hidden Markov model. In [21] and [3], convolutional neural networks and recurrent neural networks are employed to extract the features and recognize the type of the activity.

However, the features derived by these methods are then applied in a subsequent supervised model learning phase [65] because they are usually not sufficient for direct

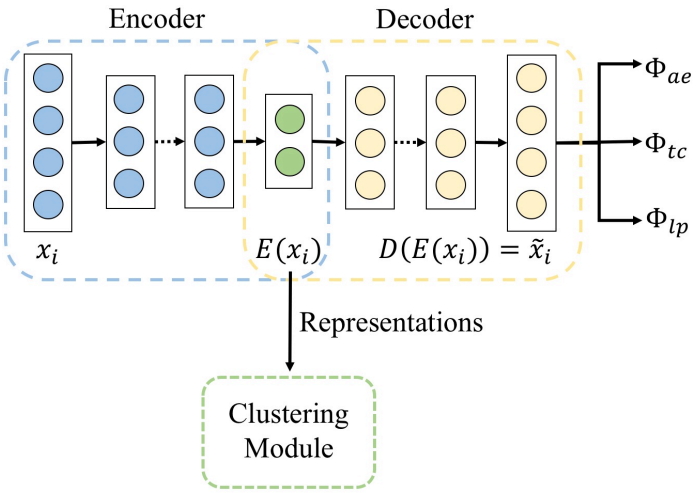


Figure 4.2: The overall architecture of the approach.

use in an unsupervised model. There are only a few works that are recognizing the activity type in an unsupervised manner. An variational AE is used in [66] to compress the sensor data into meaningful features. In [67], a protein interaction method was used to cluster the activity data. In [68], a set of time-frequency domain features are adopted, and unsupervised methods, in particular DBSCAN and mixture of Gaussian are used to cluster five basic activities.

4.3 Approach

Our approach differs from other works by using the aforementioned properties with regard to the nature of the activities. Specifically, our approach attempts to leverage two types of relationships: the temporal coherence of time series data, and locality preservation in the feature space.

4.3.1 Architecture

As shown in Fig.4.2, the foundation of the overall architecture of our approach is an AE framework. It consists of two parts: an encoder, and a decoder. The encoder

defines the transformation: $E(\cdot)$, which transforms the input data sample x_i to the representation $E(x_i)$; and the decoder defines another transformation: $D(\cdot)$, which attempts to reconstruct the original input x_i based on the its representation $E(x_i)$:

$$\tilde{x_i} = D(E(x_i))$$

where, $\tilde{x_i}$ is the reconstructed input. In the traditional AE, the loss function of one data sample can be defined as:

$$\Phi_{ae}(x_i) = ||x_i - \tilde{x_i}||^2$$

where $||\cdot||$ denotes the Euclidean distance. This loss function forces the reconstruction $\tilde{x_i}$ to be as similar as possible to the original input x_i. If a good reconstruction $\tilde{x_i}$ can be decoded from the representation $E(x_i)$, it means the representation $E(x_i)$ has retained much of the information that is important in the input x_i, so that the reconstruction $\tilde{x_i}$ can be very similar to the original input x_i. Thus the representation $E(x_i)$ can be used in other tasks, such as classification or clustering.

However, merely retaining information for reconstruction is usually not enough. The aim of the traditional AE is to learn a representation $E(x_i)$ that contains sufficient information to reconstruct the input, so an exact reconstruction also means to reconstruct noise and all the details in the input data. But not all the information in the learned representation is relevant to the subsequent task (e.g. noise as well as some task-irrelevant details might not only be unnecessary but could even be detrimental, especially in the context of subsequent clustering). Therefore, more task-oriented loss functions are imposed in our approach to guide the learning of the AE and make the learned representations more useful in the subsequent clustering task.

4.3.2 Temporal Coherence

Intuitively, a human activity can be decomposed into two components, a temporally varying component and a temporally stationary component. Specifically, certain dynamic properties of a single activity can vary over the time. For example, while walking the body pose varies over time: left foot and right foot alternatively step forward. This type of dynamic property is recorded in the sensor data too, and we refer it here as the temporally varying component. On the other hand, no matter how the body pose varies over the time, the semantic content of the activity remains the same. Namely, left foot and right foot can step forward alternatively, but the type of the activity is still walking. We refer this part as the temporally stationary component.

Based on this nature of the human activity, the temporal coherence loss forces temporally close data samples to be similar to one another, and ignore the difference in the temporal varying component. It is motivated by the intention that the semantic content, i.e. the type of the activity, in which we are interested, should vary relatively infrequently over time. If the data samples are temporally close to each other, they may represent the same type of activity, even as they may be very distant in terms of the Euclidean distance in the sensor data space. The temporal coherence loss preserves the temporal continuity of the sensor data.

More formally, let x_i^t denote a data sample i, which occurs at time t during the course of an activity. Let M_i^t denote the index set of m temporal neighbors, x_j, of x_i^t. Then the temporal coherence loss Φ_{tc} for x_i^t is defined as:

$$\Phi_{tc}(x_i^t) = \frac{1}{m} \sum_{j \in M_i^t} ||x_j - \tilde{x_i^t}||^2$$

The temporal coherence loss encourages the reconstruction $\tilde{x_i^t}$ to be similar to its temporal neighbors so that the encoder can extract useful features from the temporally stationary component and ignore irrelevant time varying details.

4.3.3 Locality Preservation

Locality preservation is inspired by the observation that different persons perform the same type of activity in different fashions, but different fashions don't hinder other people to identify the activity type. Hence we assume that the personal or individual features in the activity data may not be necessary in the activity clustering stage, and the features, which are commonly present across multiple data points, may be the essential features of the activity. The locality preserving loss function is based on this assumption.

In past research works, the combination of carefully designed handcrafted high level features to represent the main characteristics of a temporally varying signal value, and of the k-Nearest Neighbor algorithm proved to be a powerful method to classify the sensor data of human activities [55]. Due to its effectiveness and simplicity, it is employed in this approach to define the local neighborhood of a data sample. The locality preserving loss then aims to preserve the high level feature characteristics that are generally present in the local neighborhood.

The locality preserving loss forces the decoder to decode a data sample by using the learned representation of its nearby data samples. The rationale here is that if the data samples are close to each other in the handcrafted feature space, they may represent the same type of activity. Thus the features shared across multiple nearby data samples should be the essential features of that type of activity. If the features do not exist in all the nearby data samples, then the features may represent personal or individual features, but not activity features.

Formally, let x_i^f denote data sample i in the feature space, and $\tilde{x_i^f}$ the reconstruction of x_i^f. Let N_i^f denote the index set of n local neighbors, x_k, of x_i^f in the handcrafted feature space. Then the locality preserving loss Φ_{lp} for x_i^f is defined as:

$$\Phi_{lp}(x_i^f) = \frac{1}{n} \sum_{k \in N_i^f} ||x_k - \tilde{x_i^f}||^2$$

The locality preserving loss forces the model to recover data sample x_k with the representation of its nearby point x_i^f. It drives the encoder to encode the information that has generally occurred across the neighborhood and to disregard individual features of single samples.

4.3.4 Joint Loss Function

The joint loss function is the sum of the temporal coherence loss and the locality preserving loss. It is used to train the model and is defined as follows:

$$min \sum_{i=1}^{S} (1 - \alpha - \beta)\Phi_{ae}(x_i) + \alpha\Phi_{tc}(x_i^t) + \beta\Phi_{lp}(x_i^f)$$

where i is the index of the sample, S is the size of the dataset, α and β are the parameters to balance the contribution of Φ_{ae}, Φ_{tc}, and Φ_{lp}. While Φ_{tc} and Φ_{lp} preserve more task relevant information in the representation, the Φ_{ae} component is also necessary in the learning process because without the reconstruction loss Φ_{ae}, the risk of learning trivial solutions or worse representations will be increased [69].

4.3.5 Feature Extraction

After the description of the proposed model, this section focuses on the features used in the experiments. In the feature extraction stage, the segmented raw sensor signals are converted into the feature vectors. Formally, let r_i denote the sample i in

Table 4.1: List of the used statistical features.

Feature extraction function	**Description**
$mean(r_i) = \frac{1}{N}\sum_{j=1}^{N} r_{i_j}$	Mean
$var(r_i) = \frac{1}{N}\sum_{j=1}^{N}(r_{i_j} - mean(r_i))^2$	Variance
$std(r_i) = \sqrt{var(r_i)}$	Standard deviation
$median(r_i)$	Median values
$max(r_i)$	Largest values in array
$min(r_i)$	Smallest value in array
$iqr(r_i) = Q_3(r_i) - Q_1(r_i)$	Interquartile range

the set of the segmented raw sensor signals, x_i the converted feature vector, C the feature extraction function. Then the feature extraction can be defined as:

$$x_i = C(r_i)$$

The x_i is used as the input to the proposed model. Table 4.1 illustrates the statistical high level features that are used in the proposed approach. Mean, variance, standard deviation, median, which are the most commonly adopted features in the HAR research works, are used in the approach. In addition, some other features, which have been shown to be efficient in previous works [15], are included here as well. For example, the feature interquartile range (iqr), Quartiles (Q_1, Q_2 and Q_3) divide the time series signal into quarters. And iqr is the measure of variability between the upper and lower quartiles, $iqr = Q_3 - Q_1$.

All these features are computed for each axis separately. Since the data from different sensors is synchronized, combining different sensor data is achievable. In the training process, the proposed model takes these derived features as input and learns to retain the task-relevant information in the features, and to disregard the unnecessary task-irrelevant parts.

4.3.6 Cluster Construction

To train the network, the standard backpropagation algorithm with stochastic gradient descent is used. All the weights are initialized to small values and the network is evaluated on the validation data after each epoch. When the validation error stops decreasing for a predefined number of epochs, the training process is complete.

After the learning process to establish the hidden representation within the AE architecture, the trained model (i.e. the encoder of the architecture) can project the input data sample x into a clustering-friendly embedding space. More specifically, with the temporal coherence loss and the locality preserving loss, the encoder in the model is learned to encode the essential features across multiple data samples and disregard individual or temporal details that are irrelevant to the clustering task. The learned representations are evaluated in the subsequent clustering task. k-means (KM), which is arguably the most popular clustering algorithm, is used in the experiments. The number of clusters is chosen to be the true number of classes in each dataset.

4.4 Evaluation and Experiments

To evaluate the proposed network, three publicly available benchmark datasets, which contain wearable sensor data of different human activities, are used in the experiments to verify the effectiveness of the approach. The architecture of our approach for the three datasets is summarized in Table 4.2. The activation function used in the model is LeakyReLu.

Table 4.2: The architecture of our approach for different datasets. Here, only the architecture of the encoder is shown. The decoder reverses the encoder.

Dataset	**The number of neurons in each layer**
PAMAP2	Input - 128 - 64
REALDISP	Input - 256 - 128
SBHAR	Input - 30 - 20

4.4.1 Datasets

The three HAR datasets used here are PAMAP2 [55], REALDISP [70], and SBHAR [36]. We use five-fold cross-validation to measure performance and all the sensor data sequences are segmented with the sliding window method.

PAMAP2 is collected from 9 participants performing 12 activities using 3 inertial measurement units placed on the wrist, chest and ankle. The dataset contains data of sport exercises (rope jumping, nordic walking etc.), and household activities (vacuum cleaning, ironing etc.). During the experiments, heart rate, accelerometer, gyroscope, magnetometer, and temperature data is recorded. In accordance with previous research on this dataset, a sliding window of 5.12 seconds with one second step size is used to segment the data.

REALDISP is recorded from 17 volunteers carrying out 33 activities using 9 sensors placed on both arms, both legs, and the back. Each sensor provides acceleration, gyroscope, magnetic field orientation and quaternions. This dataset contains data of fitness exercises, warm up, and cool down. The sliding window used here has a size of 2 seconds without overlapping.

SBHAR is gathered from 30 participants performing 6 basic activities, such as walking, lying, and 6 postural transitions, such as sit-to-lie, sit-to-stand. In our experiments, all the postural transitions are treated as one general transition. The dataset was collected by using a smartphone placed on the waist of the participants.

The data is segmented using a sliding window of 2.56 seconds with a step size of 1.28 seconds.

4.4.2 Validation Metrics

Three evaluation metrics are adopted to measure the performance of the approach: clustering accuracy (ACC), adjusted Rand index (ARI), and normalized mutual information (NMI). The ARI and NMI are computed as follows:

$$\text{ARI} = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{n_i}{2}\sum_j \binom{n_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{n_i}{2} + \sum_j \binom{n_j}{2}] - [\sum_i \binom{n_i}{2}\sum_j \binom{n_j}{2}]/\binom{n}{2}}$$

$$\text{NMI} = \frac{\sum_i \sum_j n_{ij}\log(\frac{n \cdot n_{ij}}{n_i \cdot n_j})}{\sqrt{\sum_i n_i \log\frac{n_i}{n} \sum_j n_j \log\frac{n_j}{n}}}$$

where n_{ij} is the number of samples in cluster i and class j, n_i is the number of samples in cluster i formed using the unsupervised appraoch, n_j is the number of samples in class j as indicated by the labels in the dataset, and n is the number of samples.

4.4.3 Results and Analysis

Our proposed approach is a domain-specific extension based on traditional AE, so we compare the performance of the proposed method with the traditional AE and principal components analysis (PCA). The results of the experiments are summarized in Table 4.3.

As shown in the table, our approach achieves improved results over all three datasets. KM is applied in the embedding space. The number of clusters is chosen manually, and different cluster numbers are tested. The true number of classes in each dataset is used as the basis: tn. The results show that our approach can derive meaningful features to the subsequent activity clustering tasks. Figure. 4.3 and

4.4 also show and compare the performance of the traditional AE and the proposed approach by means of confusion matrices.

Table 4.3: The comparison between the proposed unsupervised approach and other unsupervised methods on the wearable sensor-based human activity datasets.

Methods	ACC	ARI	NMI
PAMAP2			
PCA + KM	0.6993	0.6440	0.7905
AE + KM	0.7706	0.6862	0.7994
Proposed Method + KM (tn)	0.8543	0.8016	0.8730
Proposed Method + KM ($tn+1$)	0.8622	0.8089	0.8898
Proposed Method + KM ($tn+2$)	**0.9211**	0.8288	0.8909
Proposed Method + KM ($tn+3$)	0.9150	**0.8590**	**0.9144**
REALDISP			
PCA + KM	0.5723	0.4035	0.6890
AE + KM	0.6401	0.5446	0.7764
Proposed Method + KM (tn)	0.6812	0.6051	0.8043
Proposed Method + KM ($tn+1$)	0.6829	0.6062	0.7965
Proposed Method + KM ($tn+2$)	0.7057	0.6349	0.8215
Proposed Method + KM ($tn+3$)	**0.7149**	**0.6508**	**0.8282**
SBHAR			
PCA + KM	0.6589	0.5784	0.7194
AE + KM	0.6369	0.5090	0.7048
Proposed Method + KM (tn)	0.7401	0.6343	0.7569
Proposed Method + KM ($tn+1$)	0.7596	**0.6718**	**0.7982**
Proposed Method + KM ($tn+2$)	0.8018	0.6548	0.7552
Proposed Method + KM ($tn+3$)	**0.8073**	0.6645	0.7652

In addition, Table 4.4 shows the comparison between the best results from our unsupervised approach and the results from previous published supervised methods on these three datasets. Note that, because ARI and NMI are metrics used to measure the performance of clustering algorithms, only ACC is adopted here to compare the results. As shown in the table, supervised methods can still achieve much better performance than unsupervised methods, but, as discussed before, the labeled data is usually difficult to acquire.

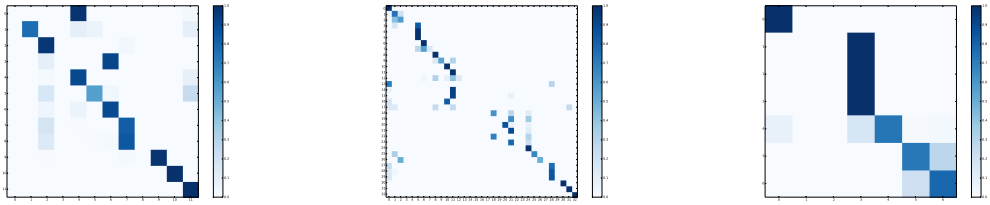
The results of the experiments show the efficiency of the approach, but we also noticed some inaccuracies introduced by this approach. One problem is locality

Table 4.4: The comparison between the proposed unsupervised approach and other supervised methods on the wearable sensor-based human activity datasets.

Methods	ACC
PAMAP2	
Probability SVM with Filter [15]	0.9304
Decision Tree (C4.5) [71]	0.9709
Boosted C4.5 [71]	**0.9980**
Proposed Method + KM ($tn + 2$)	0.9211
REALDISP	
Probability SVM with Filter [15]	**0.9952**
kNN [70]	0.9600
Proposed Method + KM ($tn + 3$)	0.7149
SBHAR	
Probability SVM with Filter [15]	0.9678
CNN [41]	**0.9870**
Proposed Method + KM ($tn + 3$)	0.8073

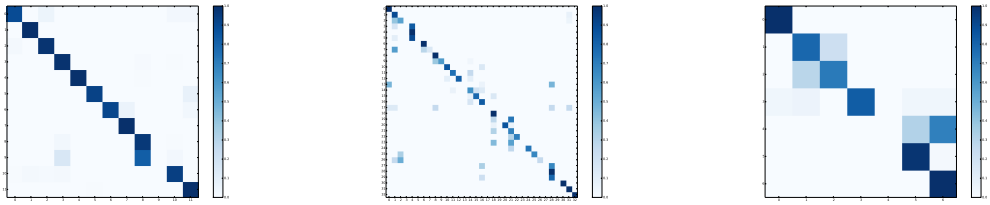
preserving loss will mix some similar activities. For example, the activities jogging and running are located closely in the embedding space. The possible reason is that the difference between jogging and running is subtle. Jogging can be seen as a slow form of running. Moreover, different people jog or run at different speeds. Thus, the locality preserving loss can drive the model to project these two different activities to adjacent locations in the embedding space. Another problem is that when a person switches between different activities, the temporal coherence assumption does not hold. During the activity transition process, the temporally adjacent data samples may represent different types of activities, hence the temporal coherence assumption will introduce inaccuracy into the model.

However, the problems mentioned above are usually infrequent. Therefore, the proposed approach can still learn useful representations and boost the performance of the subsequent clustering algorithm.



(a): PAMAP2 (b): REALDISP (c): SBHAR

Figure 4.3: Confusion Matrices of Traditional AE on PAMAP2, REALDISP, and SBHAR.



(a): PAMAP2 (b): REALDISP (c): SBHAR

Figure 4.4: Confusion Matrices of Proposed Method on PAMAP2, REALDISP, and SBHAR.

4.4.4 Ablation Studies

To further understand the effect of the different loss terms, a set of ablation experiments are conducted to measure the influences of the different loss terms separately. In the ablation experiments, the original joint loss function is transformed into two separate objectives: (i) the temporal coherence (TC) loss with the AE reconstruction loss; (ii) the locality preservation (LP) loss with the AE reconstruction loss. Each objective is used to train the model separately, thus comparing their results should provide insight into the benefit of each loss term. The results of the ablation studies are listed in Table 4.5.

We notice that including both loss terms in the objective function can clearly improve the clustering performance. These results suggest that both loss terms guided

Table 4.5: Ablation studies on the effect of each loss term.

Methods	ACC	ARI	NMI
PAMAP2			
TC Loss + AE Loss	0.7859	0.7253	0.8140
LP Loss + AE Loss	0.8065	0.7493	0.8337
Joint Loss	**0.8543**	**0.8016**	**0.8730**
REALDISP			
TC Loss + AE Loss	0.6341	0.5348	0.7639
LP Loss + AE Loss	0.6610	0.5890	0.7829
Joint Loss	**0.6812**	**0.6051**	**0.8043**
SBHAR			
TC Loss + AE Loss	0.6449	0.5173	0.7087
LP Loss + AE Loss	0.7308	0.6182	0.7440
Joint Loss	**0.7401**	**0.6343**	**0.7569**

the model to capture different useful information in the human activity sensor data in an unsupervised manner.

4.5 Conclusions

In this work, we have presented an unsupervised embedding learning approach, which is based on an autoencoder framework and uses the properties of human activities: temporal coherence and locality preservation, to project the activity data into the embedding space. We have demonstrated the effectiveness of the approach by applying it to three widely used HAR benchmark datasets. The results of the experiments show that our approach can group similar activities together in the embedding space and therefore help improve the performance of the subsequent clustering task.

CHAPTER 5

Knowledge-Guided Human Activity Recognition with Limited Data

5.1 Introduction

Many different machine learning techniques have been studied to address various problems in wearable based human activity analysis. While most of them employed supervised approaches, hence heavily relying on labeled data, very limited works have been proposed to solve the recognition problem in an unsupervised or semi-supervised way. However, due to the fact that labeled data is very scarce and the majority of the collected data has no labels, using supervised methods to recognize various human activities remains a challenging research problem in pervasive computing. On the other hand, unsupervised methods don't need labeled data to train the model, yet, the performance of unsupervised methods are usually inferior to supervised methods.

To address the aforementioned challenges we aim at developing an approach that can leverage the domain knowledge of human activity, a limited amount of similarity information of data samples, and a huge amount of unlabeled data. The proposed approach is motivated by two factors: first, using domain knowledge and a large amount of unlabeled data to simplify the learning task of the model and force the model to retain useful task-specific features, and second, limiting the supervision needed for training the model by only using a very small amount of labeled data and only using the similarity of the data samples, but not using the labels explicitly. Experiments have been conducted to evaluate the effectiveness of our proposed approach.

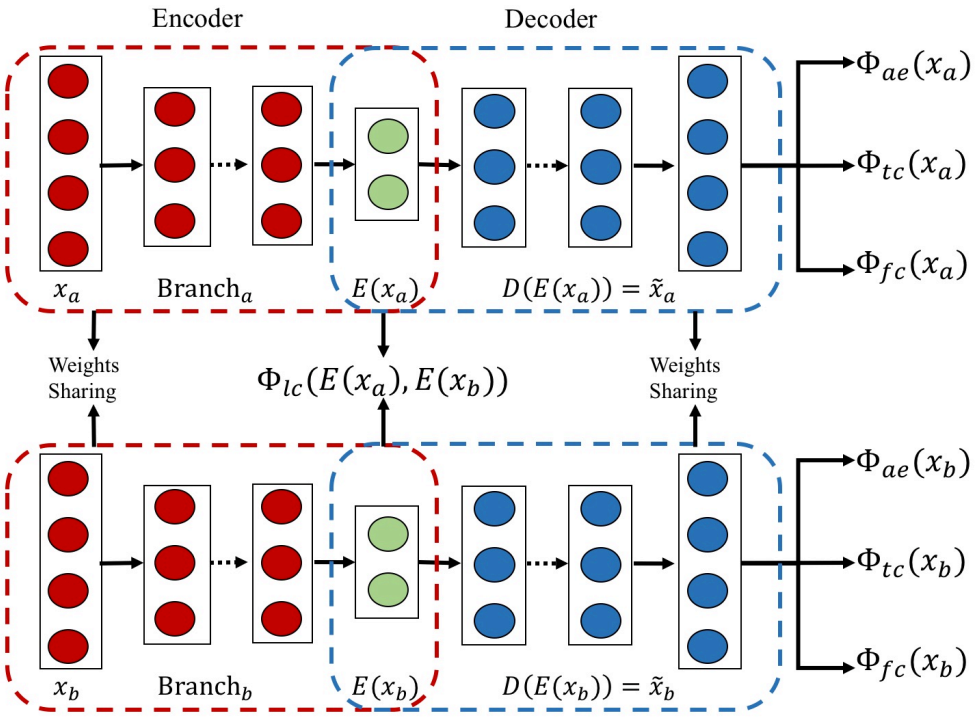


Figure 5.1: The overall architecture of the proposed approach.

5.2 Proposed Approach

In this section, we first introduce the architecture of the proposed model. Then we show the definitions of various consistency criteria that are used in the model, and how the domain knowledge is employed in the model in the form of these consistencies.

5.2.1 Architecture of the Proposed Model

As shown in Fig. 5.1, the foundation of the overall architecture of this approach is a combination of an autoencoder and siamese networks.

Autoencoder is a powerful, widely used unsupervised deep architecture for feature learning. It consists of two parts: an encoder, and a decoder. The encoder defines the nonlinear transformation: $E(\cdot)$ that encodes the input data sample x into the representation $E(x)$. The decoder defines another nonlinear transformation $D(\cdot)$

that aims at decoding the representation $E(x)$ and reconstructing the original input x.

$$\tilde{x} = D(E(x))$$

Here, x is the input data, $E(x)$ is the encoded representation from the encoder, and $\tilde{x}$ is the decoded reconstruction from the decoder. The learning process of an autoencoder is to minimize the reconstruction loss:

$$\Phi_{ae}(x) = \sum_{x \in X} ||x - \tilde{x}||^2$$

where X denotes the dataset. This loss function forces the reconstruction $\tilde{x}$ to be as similar as possible to the original input x. If a good reconstruction $\tilde{x}$ can be decoded from the representation $E(x)$, it means the representation $E(x)$ has retained much of the information that is important in the input x, so that the reconstruction $\tilde{x}$ can be very similar to the original input x. Thus the representation $E(x)$ can be used in other tasks, such as classification or clustering.

Siamese Networks, as illustrated in Figure 5.1, are dual-branch neural networks with shared weights such that $Branch_a = Branch_b$. Given an input pair of human activity data samples $\{x_a, x_b\}$, the siamese network learns to project the input pair to the representation pair $\{E(x_a), E(x_b)\}$. The distance between the representation pair is then used as the semantic similarity of the input pair.

To learn a good representation for correctly classifying activities, however, merely retaining information for reconstruction is usually not enough. Instead, the representation should represent relevant information for the task while largely ignoring irrelevant aspects. The aim of an autoencoder is to learn a representation $E(x)$ that contains sufficient information to reconstruct the input, so an exact reconstruction also means to reconstruct noise and all the details in the input data. But not all the information in the learned representation is relevant to the subsequent task (e.g.

noise as well as some task-irrelevant details might not only be unnecessary but could even be detrimental, especially in the context of subsequent clustering). Therefore, more task-oriented loss functions are needed to be imposed on our model to guide the learning of the autoencoder and make the learned representations more useful in the subsequent clustering task. To allow this in a semi-supervised setting, the proposed architecture utilizes unsupervised consistency criteria within the autoencoder components as well as weakly-supervised criteria through the siamese network structure. These consistency criteria complement each other, permitting high accuracy activity recognition with only limited amounts of weakly supervised data.

5.2.2 Consistency Definition

The consistencies employed in the model are the common sense knowledge or domain knowledge, which can provide useful task-oriented information to the model, guide the model to learn task-relevant information, and ignore task-irrelevant information, hence improving the performance of the model.

Temporal Consistency: Let x_i^t denote the i-th data sample that occurs at time t during the course of an activity. Let X_i^t denote the set of m data samples which are the nearest neighbors of x_i^t based on the temporal closeness during the course of an activity, and $\tilde{x_i^t}$ the reconstruction of x_i^t.

$$\Phi_{tc}(x_i^t) = \frac{1}{m} \sum_{x_j \in X_i^t} ||x_j - \tilde{x_i^t}||^2$$

Temporal consistency attempts to structure data such that data samples that occur directly after each other are closer together in representation space, representing the intuition that activities are usually longer than a single data sample and thus that consecutive data samples are often still part of the same activity.

Feature Consistency: Let x_i^f denote the i-th data sample in feature space f, and X_i^f the set of n data samples which are the nearest neighbors of x_i^f in the feature space f, and $\tilde{x_i^f}$ the reconstruction of x_i^f.

$$\Phi_{fc}(x_i^f) = \frac{1}{n} \sum_{x_k \in X_i^f} ||x_k - \tilde{x_i^f}||^2$$

Feature consistency aims to keep data samples that have very similar features close together in the embedding space. This reflects the intuition that data segments that exhibit very similar sensor readings are more likely to belong to the same activity.

Label Consistency: With the help of siamese networks and a very small amount of labeled data, a group of pairwise constraints can be defined on the input data. This group of pairwise constraints can drive the proposed model to map the data into a representation space, in which the clusters satisfy the given constraints. More formally, given an input pair of human activity data sequence $\{x_a, x_b\}$, the encoder of the siamese networks learns to encode the input pair to the representation pair $\{E(x_a), E(x_b)\}$. Then, a distance function, which measures the similarity between the encoded representations, is calculated.

$$\Phi_{lc}(x_a, x_b) = ||E(x_a) - E(x_b)||^2$$

The similarity distance $Dist(x_a, x_b)$ indicates if the input pair represents the same type of activity. A large distance indicates two different types of activities, while a small distance indicates the same types of activity.

The aforementioned three types of consistency criteria imposed on the proposed model enable the learning to capture the information that is useful to the activity recognition task.

5.2.3 Joint Loss Function

The training process of the model is divided into two stages. In the first stage the model is trained with the following joint loss function:

$$min \sum_{i=1}^{S} (1 - \alpha - \beta) \cdot \Phi_{ae}(x_i) + \alpha \cdot \Phi_{tc}(x_i^t) + \beta \cdot \Phi_{fc}(x_i^f)$$

where i is the index of the sample, S is the size of the dataset, α and β are the parameters to balance the contribution of Φ_{ae}, Φ_{tc}, and Φ_{fc}. While Φ_{tc} and Φ_{fc} retain task-relevant features and disregard the unnecessary task-irrelevant features, the Φ_{ae} component is also necessary in the learning process because without the reconstruction loss Φ_{ae}, the risk of learning trivial solutions or worse representations will be increased [69]. This represents and initial unsupervised training phase in which the embedding space is trained to reflect the basic structure of the data.

In the second training stage, the label consistency loss Φ_{lc} is added to further improve the performance of the model. Since in the first stage, most clusters are already formed, the second stage imposes a small amount of pairwise constraints on the data using the limited amount of available weakly supervised data to readjust the already formed clusters to adapt to the available information about the intended classes. The joint loss function is defined as follow:

$$\begin{aligned} &min \sum_{i=1}^{S} (1 - \alpha - \beta - \gamma) \cdot (\Phi_{ae}(x_a) + \Phi_{ae}(x_b)) + \\ &\alpha \cdot (\Phi_{tc}(x_a^t) + \Phi_{tc}(x_b^t)) + \beta \cdot (\Phi_{fc}(x_a^f) + \Phi_{fc}(x_b^f)) + \gamma \cdot \Phi_{lc}(x_a, x_b) \end{aligned}$$

The label consistency supervision Φ_{lc} used in the model is limited, but it is directly related to the activity recognition task, so in the second stage, the label consistency will have a larger weight and dominate the learning process, and the other loss functions will have smaller weights, so that the model can learn directly task-related features. In addition, the amount of label consistency supervision used in the model is very

limited, therefore, the other 3 unsupervised loss functions are still needed to work as regularization terms to prevent potential overfitting.

5.2.4 Feature Extraction

After the description of the proposed model, this section focuses on the features used in the experiments. In the feature extraction stage, the segmented raw sensor signals are converted into the feature vectors. Formally, let r_i denote the sample i in the set of the segmented raw sensor signals, x_i the converted feature vector, C the feature extraction function. Then the feature extraction can be defined as:

$$x_i = C(r_i)$$

The x_i is used as the input to the proposed model. Table 5.1 illustrates the statistical high level features that are used in the proposed approach. Mean, variance, standard deviation, median, which are the most commonly adopted features in the HAR research works, are used in the approach. In addition, some other features, which have been shown to be efficient in previous works [15], are included here as well. For example, the feature interquartile range (iqr), which is based on Quartiles (Q_1, Q_2 and Q_3) that divide the time series signal into quarters. Using these, iqr is the measure of variability between the upper and lower quartiles, $iqr = Q_3 - Q_1$.

All these features are computed for each axis separately. Since the data from different sensors is synchronized, combining different sensor data is achievable. In the training process, the proposed model takes these derived features as input and learns to retain the task-relevant information in the features, and to disregard the unnecessary task-irrelevant parts.

Table 5.1: List of the used statistical features.

Feature extraction function	**Description**
$mean(r_i) = \frac{1}{N}\sum_{j=1}^{N} r_{i_j}$	Mean
$var(r_i) = \frac{1}{N}\sum_{j=1}^{N}(r_{i_j} - mean(r_i))^2$	Variance
$std(r_i) = \sqrt{var(r_i)}$	Standard deviation
$median(r_i)$	Median values
$max(r_i)$	Largest values in array
$min(r_i)$	Smallest value in array
$iqr(r_i) = Q_3(r_i) - Q_1(r_i)$	Interquartile range

5.2.5 Cluster Construction

The standard backpropagation algorithm with stochastic gradient descent is used to train the network. All the weights are initialized to small values and the network is evaluated on the validation data after each epoch. When the validation error stops decreasing for a predefined number of epochs, the training process is complete.

After the two stage training process to establish the hidden representation within the siamese and autoencoder architecture, the trained model (i.e. the encoder of the architecture) can project the input data sample x into a clustering-friendly embedding space. The learned representations are evaluated in the subsequent clustering task. k-means (KM), which is arguably the most popular clustering algorithm, is used in the experiments. The number of clusters is chosen to be the true number of classes in each dataset.

5.3 Evaluations and Experiments

5.3.1 Datasets

We evaluate the performance of the model on dataset PAMAP2 [55]. And all the sensor data sequences are segmented with the sliding window method.

PAMAP2 is a dataset collected from 9 participants performing 12 activities using 3 inertial measurement units placed on the wrist, chest and ankle. The dataset contains data of sport exercises (rope jumping, nordic walking etc.), and household activities (vacuum cleaning, ironing etc.). During the experiments, heart rate, accelerometer, gyroscope, magnetometer, and temperature data is recorded. In accordance with previous research on this dataset, a sliding window of 5.12 seconds with one second step size is used to segment the data.

To evaluate the performance of the weakly semi-supervised architecture, it was applied with 1%, 5%, and 10% weakly supervised data and its performance was compared to fully supervised methods as well as to the unsupervised approach from Chapter 4. In all experiments, the total dataset (i.e. unlabeled and labeled) comprised all data segments of the PAMAP2 dataset, giving the supervised comparison approaches access to all the labels while the weakly semi-supervised technique has access only to similarity information for a small subset. To apply the weakly semi-supervised method, the set of data segments were split according to the above-mentioned percentage and labels were removed from the samples in the unlabeled data set. The small percentage of labeled data was then converted into weakly supervised data pairs which only contain information whether they belonged to the same or a different class.

5.3.2 Results and Analysis

As shown in result table 5.2, the proposed weakly semi-supervised approach achieved significant improvement over the purely unsupervised method with the same number of clusters even with very limited data and improved performance with the increase of available weakly supervised data. Moreover, performance was reasonable good even with only one percent of the labeled training data. With only 5% of weakly supervised data, performance of the model exceeded all but one of the fully

Table 5.2: Results on PAMAP2

Methods	**Activity**
LSTM-F [4]	0.9290
CNN [4]	0.9370
DNN [4]	0.9040
Probability SVM with Filter[15]	0.9304
Decision Tree (C 4.5)	0.9709
Boosted C 4.5 [55]	**0.9969**
Unsupervised Learner in chapter 4 + KM (TN)	0.8543
Unsupervised Learner in chapter 4 + KM (TN+2)	0.9211
Proposed weakly semi-supervised model + 1% training	0.9128
Proposed weakly semi-supervised model + 5% training	0.9755
Proposed weakly semi-supervised model + 10% training	0.9904

supervised comparison methods, and with 10% of the labeled data, the proposed approach can achieve competitive performance compared with even the best of the purely supervised methods, despite it having access to only a fraction of the labeled data.

5.4 Conclusions

In this work, we have presented a weakly semi-supervised embedding learning approach, which is based on an autoencoder architecture and a siamese architecture. The proposed approach uses the properties of human activities: temporal consistency and feature consistency to project the activity data into the embedding space, then imposes a small amount of pairwise constraints on the data to fine-tune the model in a weakly supervised fashion. We have demonstrated the effectiveness of the approach by applying it to a widely used HAR benchmark datasets. The results of the experiments show that with only ten percent of the labeled data, the proposed approach can achieve competitive result, which significantly reduced the supervision needed to train the recognition model.

CHAPTER 6

Learning Embeddings for New Activities Using Hierarchical Human Activity Modeling

6.1 Introduction

The recognition of human activities is a crucial component in many user-centric application, such as assistive technology, user behavior model, and health management. There has been extensive research on wearable-based human activity recognition system. A typical recognition system is built to recognize the activities that were previously observed in the training data. However, in many real-world application scenarios, the recognition system needs to recognize new unobserved activities. One common solution for this problem is to ask users to label additional data samples of the new unobserved activities, and retrain the system on these newly labeled data. Nonetheless, due to the fact that labeling data samples is usually time consuming and expensive, it is impractical to expect to get enough labeled data from the users. Thus, a recognition system that can learn new activities with a small amount of labeled data is needed. The major problems of such a system are: (i) With very limited data, the recognition model is prone to overfit to the new activity data samples. (ii) If the new activity has only a few training samples, the recognition model tends to underestimate the probability of the occurrence of the new activity in the dataset, therefore, the new activity will not be correctly predicted.

To address the aforementioned problems, we proposed an approach that learns the hierarchical structure of the human activity, and based on the hierarchical structure, a high-level activity can be decomposed into a series of mid-level primitive

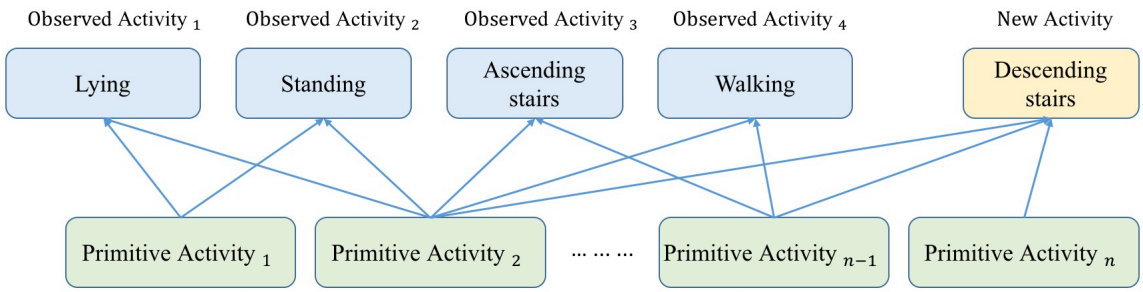


Figure 6.1: Hierarchical Human Activity Modeling

activities. Therefore, even if the high-level activity is new and unobserved to the recognition system, it can still share the same types of mid-level primitive activities with the previously observed high-level activities. Figure 6.1 shows how a high-level activity can be decomposed into multiple mid-level primitive activities. We assume that the primitive activity describes an intrinsic property or a basic characteristic of a high-level activity, and intrinsic properties or basic characteristics are generally present in different high-level activities. Therefore, a new unobserved activity can be described with the primitive activities that are extracted from other observed activities. Experiments have been conducted to evaluate the effectiveness of the proposed approach.

6.2 Proposed Approach

Essentially, human activity embedding with limited amounts of data faces four great challenges to achieve a good embedding result:

- Insufficient data: the amount of data samples for the new unobserved activity is limited.
- Imbalanced data: while training data for the new activity is limited, the amount of training data for all the other previously observed activities is significantly larger.

- Need for mid-level activites: Most previous works [72, 73] on zero-shot or few-shot learning with sequential sensor data focused on using well-established semantic attributes (such as arm swinging, leg still, body medium motion). On the one hand, the semantic attributes provide effective guidance to the model training, but on the other hand, employing semantic attributes requires additional annotation work.
- Highly non-linear time series data: usually, the activity data is collected from multiple sensors that record the movement of different body parts over time. This data is highly non-linear, thus it is difficult to capture the non-linearity.

To address these challenges, we developed a new approach that decomposes high-level activities into a series of mid-level primitive activities. The architecture of the proposed model is illustrated in Figure 6.2. In this approach, feature vectors for primitive activities are learned in an unsupervised fashion using an autoencoder framework to reduce the need for labeling semantic properties in the data. These are then concatenated to present the input space for the high-level activity recognition system.

6.2.1 Insufficient and Imbalanced Data

To mitigate the insufficient and imbalanced data limitation in the context of a new activity, we avoid to train the activity model on the high-level new activity, which does not have enough training data. Instead, we proposed to decompose high-level activities into a series of mid-level primitive activities.

The amount of training data for a new activity is small, but the primitive activities, which are the components of the new activity, are commonly present in the other previously observed high-level activities. Thus, mid-level primitive activity models can be trained by using the primitive activity data from previous activities. Then the

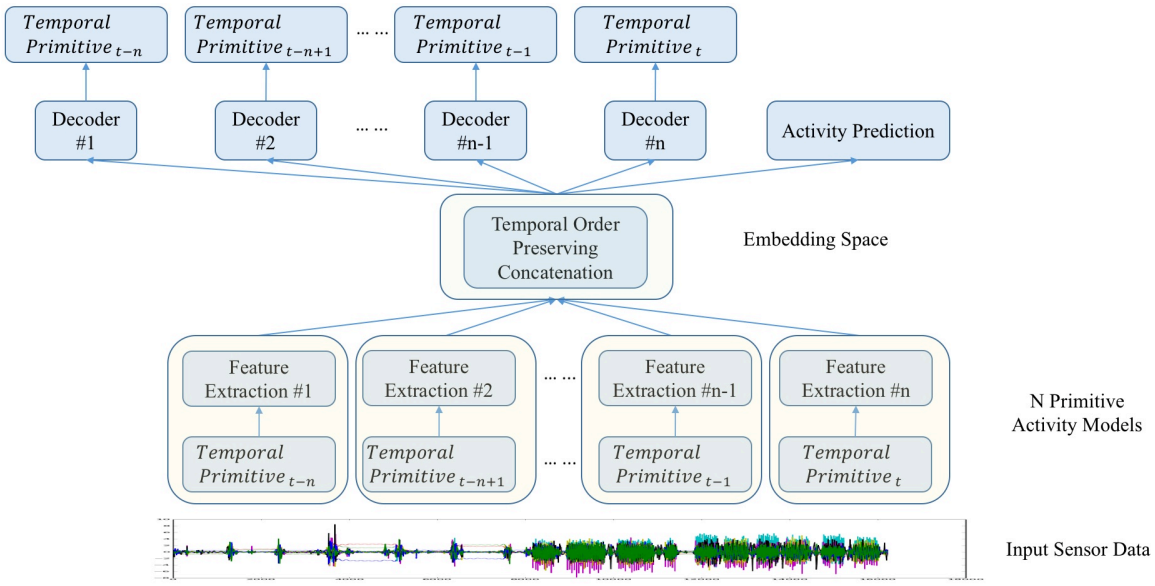


Figure 6.2: The architecture of the proposed model

trained primitive activity models can be used to represent the unobserved or newly observed activity to alleviate insufficient and imbalanced training data problems for the new activity.

6.2.2 Temporal Primitive Activity

Most previous works [72, 73, 74] rely on additional semantic attributes to label the data and reduce the burden of training the model on an unobserved activity. However, labeling the semantic attributes is also associated with additional cost. Thus, in our proposed model, we leverage the hierarchical and sequential characteristics of activity data to decompose high-level activity into a series of mid-level primitive activities. In the sequence of primitive activities, each primitive activity represents the high-level activity during a specific stage. That is to say concatenating all the primitive activities in the correct temporal order will regenerate the original high-level activity. Similarly, learning a high-level activity model can be decomposed to learning a series of mid-level primitive activity models.

Formally, let $A_T = (a_0, a_1, \cdots, a_T)$ denote a high-level activity data sequence. A_T can be decomposed into a series of n primitive activities $(P_0, P_{\frac{T}{n}}, \cdots, P_{T-\frac{T}{n}})$, where $P_t = (a_t, a_{t+1} \cdots, a_{t+\frac{T}{n}-1})$ is a primitive activity data sequence, which starts at time t with the duration length $\frac{T}{n}$. Then, instead of training the model on high-level activity A_T, we train a series of primitive activity models using the decomposed mid-level activities $(P_0, P_{\frac{T}{n}}, \cdots, P_{T-\frac{T}{n}})$. Because a high-level activity consists of a series of primitive activities, we use a concatenation of the primitive activities to represent the original high-level activity. Hence, an unobserved high-level activity can be represented by a series of observed primitive activities. In addtion, the learning models for primitive activities are trained in an unsupervised way, therefore, the proposed approach achieved (i) an unobserved high-level activity can be represented by a series of observed primitive activities. (ii) no additional labeling work is needed for the primitive activity.

6.2.3 High Non-linearity

To capture the highly non-linear structure in the data, each primitive activity model is trained as an autoencoder. The autoencoder is a widely used unsupervised deep learning model for feature extraction. It consists of two parts: an encoder, and a decoder. The encoder defines the nonlinear transformation: $E(\cdot)$ that encodes the input data sample x into the representation $E(x)$. The decoder defines another nonlinear transformation $D(\cdot)$ that aims at decoding the representation $E(x)$ and reconstructing the original input x.

$$\tilde{x} = D(E(x))$$

Here, x is the input data, $E(x)$ is the encoded representation from the encoder, and $\tilde{x}$ is the decoded reconstruction from the decoder. In order to capture the high non-

linearity in the data, more than one encoding layer can be employed in the encoder, correspondingly, the same number of decoding layers are also needed in the decoder.

$$\tilde{x} = D_n(\cdots D_1(E_n(\cdots E_1(x))))$$

The learning process of an autoencoder is to minimize a reconstruction loss:

$$\Phi_{ae}(x) = \sum_{x \in X} ||x - \tilde{x}||^2$$

where X denotes the dataset. This loss function forces the reconstruction $\tilde{x}$ to be as similar as possible to the original input x. If a good reconstruction $\tilde{x}$ can be decoded from the representation $E(x)$, it means the representation $E(x)$ has retained much of the information that is important in the input x, so that the reconstruction $\tilde{x}$ can be very similar to the original input x. Thus the representation $E(x)$ can be used in other tasks, such as classification or clustering.

The data fed to the model is the features extracted from the sensor data. In the feature extraction stage, the segmented raw sensor signals are converted into the feature vectors. Formally, let r_i denote the sample i in the set of the segmented raw sensor signals, x_i the converted feature vector, and C the feature extraction function. Then the feature extraction can be defined as:

$$x_i = C(r_i)$$

Here x_i is the input to the model. Table 6.1 illustrates the statistical features that are used in the approach. In particular, mean, variance, standard deviation, median, which are among the most commonly adopted features in the human activity recognition research works, are used here. In addition, some other features, which have been shown to be efficient in previous works [15], are included here as well. For example, the feature interquartile range (iqr), which is based on Quartiles (Q_1, Q_2 and Q_3)

Table 6.1: List of the used statistical features.

Feature extraction function	**Description**
$mean(r_i) = \frac{1}{N}\sum_{j=1}^{N} r_{i_j}$	Mean
$var(r_i) = \frac{1}{N}\sum_{j=1}^{N}(r_{i_j} - mean(r_i))^2$	Variance
$std(r_i) = \sqrt{var(r_i)}$	Standard deviation
$median(r_i)$	Median values
$max(r_i)$	Largest values in array
$min(r_i)$	Smallest value in array
$iqr(r_i) = Q_3(r_i) - Q_1(r_i)$	Interquartile range

which divide the time series signal into quarters. Using these, iqr is the measure of variability between the upper and lower quartiles, $iqr = Q_3 - Q_1$.

All these features are computed for each axis separately. Since the data from different sensors is synchronized, combining different sensor data is achievable. In the training process, the model takes these derived features as input and learns to retain useful information in the features, and to disregard the unnecessary parts.

6.2.4 Joint Activity Model

In the proposed approach, a high-level activity will be represented by a concatenation of multiple primitive activity models, as shown in Figure 6.2. Then the concatenated representation will be fed into (i) multiple decoders that decode the representation and reconstruct the original primitive activities. (ii) an activity predictor that will predict the activity type of the data.

The decoder part drives the autoencoder architecture in the model to learn effective features of the primitive activities. The predictor part uses the observed activities data to refine the primitive activity models. The goal here is to let the primitive activity models retain the features that help distinguish activity types, and disregard activity type-irrelevant features.

6.2.5 Cluster Construction

To train the network, the standard backpropagation algorithm with stochastic gradient descent is used. All the weights are initialized to small values and the network is evaluated on the validation data after each epoch. When the validation error stops decreasing for a predefined number of epochs, the training process is complete.

After the learning process, in order to establish the hidden representation within the primitive activity models, the joint trained model (i.e. the joint primitive activity models) can project the input data sample x into an embedding space where data representations from the same type of activity stay together and data representations from different types of activities stay far away from each other. More specifically, the learned representations are evaluated in the subsequent clustering task. k-means (KM), which is arguably the most popular clustering algorithm, is used in the experiments. The number of clusters is chosen to be the true number of classes in each dataset.

6.3 Evaluations and Experiments

In the training stage, we leave one activity as a new unobserved activity and the remaining activities as observed activities. For observed activities, all the data samples are used to train the model, but only a limited number of training samples from the unobserved activity are selected to test the performance of the proposed model under the zero-shot and few-shot learning setup.

6.3.1 Datasets

The performance of the model is evaluated on dataset MHEALTH [56]. And all the sensor data sequences are segmented with the sliding window method.

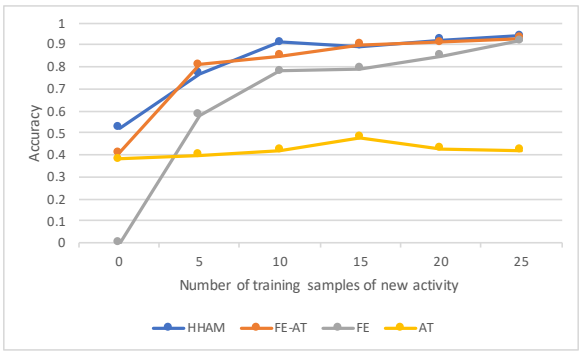


Figure 6.3: The recognition of unobserved activities on MHEALTH

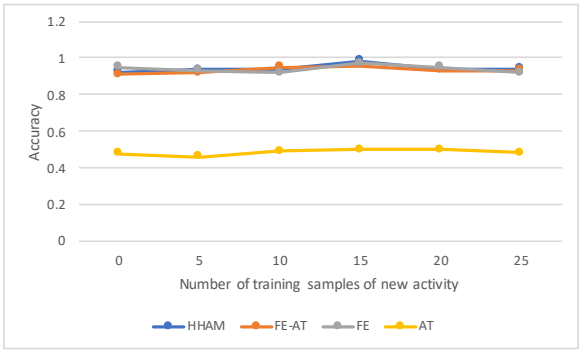


Figure 6.4: The recognition of observed activities on MHEALTH

The **MHEALTH** dataset contains data recorded from 10 volunteers carrying out 12 physical activities, including primitive body parts movements (waist bends forward, frontal elevation of arms etc), and composite body movements (cycling, jumping front and back etc). The data is collected by using three sensors placed on the subject's chest, right wrist and left ankle to record accelerometer, gyroscope, and magnetometer signals. The chest sensor also records 2-lead ECG signals. The sampling rate of all sensing modalities is 50Hz. As in previous work [60], we use a sliding window of 5 seconds with a step size of 2.5 seconds.

We compared our proposed approach with a previously published work [73], in which semantic attributes are employed. Here, HHAM represents our proposed hierarchical human activity modeling, FE represents the traditional feature-based

learning, AT represents the semantic attribute-based learning, and FE-AT is the combination of them.

Figures 6.3 and 6.4 show the average accuracy over all the leave-one-out experiments for the unseen and the seen acitivites, respectively, as a function of the number of available samples for the previously unseen activity. As shown in Figure 6.3, our proposed approach achieves comparable results for the unobserved activities. Since KM is applied in the embedding space, it shows that our approach can derive meaningful representation for the unobserved activities. In Figure 6.4, the experimental results demonstrate that the increasing amount of unobserved data samples in the training data does not affect the performance of the models on previously observed activities.

The results of the experiments show the efficiency of the approach. When no label is available, the proposed HHAM method and FE-AT method can achieve comparable performance. But in the FE-AT method, semantic attributes take the place of labels, and provide guidance to the model training. However, as we discussed, labeling semantic attributes requires additional work, while the proposed approach learns the primitive activity models in an unsupervised way and thus does not require this additional work.

6.4 Conclusions

In this work, we have presented a hierarchical human activity modeling approach, which can learn a previously unobserved new activity using limited training samples. The approach consists of a series of unsupervised primitive activity models and does therefore not rely on any additional labels or semantic attributes. The trained model maps the activity data into the an embedding space in which the subsequent classification and clustering tasks can achieve better results. We have

demonstrated the effectiveness of the approach by applying it to a widely used human activity recognition benchmark dataset. The results of the experiments show that our approach can group the same type of activities together in the embedding space, therefore helping improve the performance of the subsequent tasks including the model learning for novel high-level activities for which only a very limited number of instances are available.

CHAPTER 7

Conclusions

With the increasing availability of embedded sensors in smartphones, wearable devices and smart environments, the sensor data stream of human activity is more accessible. In order to extract effective information from the wearable-sensor data and use it to build better ubiquitous computing applications, we focus on developing various learning models that can improve the effectiveness of the wearable-sensor human activity analysis.

7.1 Contributions

The presented work used different learning models to analyze the wearable human activity data. In Chapter 2, we apply a weakly supervised method that can learn to dynamic segment sensor time-series data and recognize human activity under limited supervision. The proposed method uses only the information about the similarity between pairs of data samples to capture temporal relations efficiently without knowing the explicit labels. In Chapter 3, a multi-task model is presented to recognize human activities and identify the person simultaneously. We investigate the scalability of the proposed method by expanding it to learn representations for additional attributes in the data along with human activity recognition and person identification. The experiments show that the proposed architecture can scale to increasing numbers of tasks with very little loss of accuracy. In addtion, further experiments on training the model with imperfect data demonstrate that the model is capable of effectively utilizing cross-task information to achieve competitive performance.

In Chapter 4, an unsupervised model is proposed, which is based on an autoencoder framework and uses the properties of human activities: temporal coherence and locality preservation to supervise the model learning, learn effective features from the data without using the labels, and project the activity data into a clustering-friendly embedding space. In Chapter 5, based on the model architecture in Chapter 4, we include limited supervision to improve the model performance. The experiment results show that with limited amount of supervision, the weakly semi-supervised model can achieve comparable performance to supervised models. In Chapter 6, leveraging the hierarchical structure of human activities, we develop a recognition model that can recognize a previously unobserved activity without the need for additional labeling of semantic attributes or subactivities.

The approaches introduced in this dissertation successfully solve various problems in human activity analysis, overcoming some of the limitations of existing methods in terms of the need for extensive data labeling, and improve the performance on many real-world applications.

REFERENCES

[1] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, "Complex human activity recognition using smartphone and wrist-worn motion sensors," *Sensors*, vol. 16, no. 4, 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/4/426

[2] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Evaluating the effects of signal segmentation on activity recognition," in *International Work-Conference on Bioinformatics and Biomedical Engineering, IWBBIO 2014*, 2014, pp. 759–765.

[3] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional neural networks for human activity recognition using mobile sensors," in *6th International Conference on Mobile Computing, Applications and Services*, Nov 2014, pp. 197–205.

[4] N. Y. Hammerla, S. Halloran, and T. Plötz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI'16. AAAI Press, 2016, pp. 1533–1540. [Online]. Available: http://dl.acm.org/citation.cfm?id=3060832.3060835

[5] F. J. O. Morales and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," in *Sensors*, 2016.

[6] I. Andrey, "Real-time human activity recognition from accelerometer data using convolutional neural networks," *Applied Soft Computing*, vol. 62, 09 2017.

[7] J. R. Kwapisz, G. Weiss, and S. A. Moore, "Cell phone-based biometric identification," 10 2010, pp. 1 – 7.

[8] A. Kale, N. Cuntoor, B. Yegnanarayana, A. N. Rajagopalan, and R. Chellappa, "Gait analysis for human identification," in *Proceedings of the 4th International Conference on Audio- and Video-based Biometric Person Authentication*, ser. AVBPA'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 706–714. [Online]. Available: http://dl.acm.org/citation.cfm?id=1762222.1762314

[9] D. Gafurov and E. Snekkenes, "Gait recognition using wearable motion recording sensors," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, no. 1, p. 415817, Jun 2009. [Online]. Available: https://doi.org/10.1155/2009/415817

[10] C. McCool, S. Marcel, A. Hadid, M. Pietikinen, P. Matejka, J. Cernock, N. Poh, J. Kittler, A. Larcher, C. Lvy, D. Matrouf, J. Bonastre, P. Tresadern, and T. Cootes, "Bi-modal person recognition on a mobile phone: Using mobile phone data," in *2012 IEEE International Conference on Multimedia and Expo Workshops*, July 2012, pp. 635–640.

[11] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural Comput.*, vol. 14, no. 4, pp. 715–770, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1162/089976602317318938

[12] T. Sheng and M. Huber, "Siamese networks for weakly supervised human activity recognition," 2019, pp. 4069–4075.

[13] T. Sheng and M. Huber, "Weakly supervised multi-task representation learning for human activity analysis using wearables," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, pp. 1 – 18, 2020.

[14] T. Sheng and M. Huber, "Unsupervised embedding learning for human activity recognition using wearable sensor data," in *FLAIRS Conference*, 2020.

[15] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomput.*, vol. 171, no. C, pp. 754–767, Jan. 2016. [Online]. Available: https://doi.org/10.1016/j.neucom.2015.07.085

[16] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Comput. Surv.*, vol. 46, no. 3, pp. 33:1–33:33, Jan. 2014. [Online]. Available: http://doi.acm.org/10.1145/2499621

[17] E. Guenterberg, S. Ostadabbas, H. Ghasemzadeh, and R. Jafari, "An automatic segmentation technique in body sensor networks based on signal energy," 04 2009.

[18] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *SIGKDD Explor. Newsl.*, vol. 12, no. 2, pp. 74–82, Mar. 2011. [Online]. Available: http://doi.acm.org/10.1145/1964897.1964918

[19] T. Tamura, M. Sekine, M. Ogawa, T. Togawa, and Y. Fukui, "Classification of acceleration waveforms during walking by wavelet transform," *Methods of information in medicine*, vol. 36, pp. 356–9, 01 1998.

[20] Z. He and L. Jin, "Activity recognition from acceleration data based on discrete consine transform and svm," in *2009 IEEE International Conference on Systems, Man and Cybernetics*, Oct 2009, pp. 5041–5044.

[21] F. J. O. Morales and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," in *Sensors*, 2016.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[23] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *Proceedings of the 6th International Conference on Neural Information Processing Systems*, ser. NIPS'93.

San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 737–744. [Online]. Available: http://dl.acm.org/citation.cfm?id=2987189.2987282

[24] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, pp. 2786–2792. [Online]. Available: http://dl.acm.org/citation.cfm?id=3016100.3016291

[25] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 539–546 vol. 1.

[26] N. Zeghidour, G. Synnaeve, N. Usunier, and E. Dupoux, "Joint learning of speaker and phonetic similarities with siamese networks," in *INTERSPEECH*, 2016.

[27] P. Neculoiu, M. Versteegh, and M. Rotaru, "Learning text similarity with siamese recurrent networks," in *Rep4NLP@ACL*, 2016.

[28] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2016.

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 448–456. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045167

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[31] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. USA: Omnipress, 2010, pp. 807–814. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104425

[32] L. Pigou, A. van den Oord, S. Dieleman, M. Van Herreweghe, and J. Dambre, "Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video," *International Journal of Computer Vision*, vol. 126, no. 2, pp. 430–439, Apr 2018. [Online]. Available: https://doi.org/10.1007/s11263-016-0957-7

[33] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 843–852. [Online]. Available: http://proceedings.mlr.press/v37/srivastava15.html

[34] B. Everitt, S. Landau, M. Leese, and D. Stahl, *Cluster analysis*, 5th ed. Wiley, 2011.

[35] M. Bächlin, D. Roggen, G. Tröster, M. Plotnik, N. Inbar, I. Maidan, T. Herman, M. Brozgol, E. Shaviv, N. Giladi, and J. Hausdorff, "Potentials of enhanced context awareness in wearable assistants for parkinson's disease patients with the freezing of gait syndrome," 09 2009, pp. 123–130.

[36] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," 01 2013.

[37] M. Zeng, H. Gao, T. Yu, O. J. Mengshoel, H. Langseth, I. Lane, and X. Liu, "Understanding and improving recurrent networks for human activity recognition by continuous attention," in *Proceedings of the 2018*

ACM International Symposium on Wearable Computers, ser. ISWC '18. New York, NY, USA: ACM, 2018, pp. 56–63. [Online]. Available: http://doi.acm.org/10.1145/3267242.3267286

[38] C. Catal, S. Tufekci, E. Pirmit, and G. Kocabag, "On the use of ensemble of classifiers for accelerometer-based activity recognition," *Appl. Soft Comput.*, vol. 37, no. C, pp. 1018–1022, Dec. 2015. [Online]. Available: https://doi.org/10.1016/j.asoc.2015.01.025

[39] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional neural networks for human activity recognition using mobile sensors," in *6th International Conference on Mobile Computing, Applications and Services*, Nov 2014, pp. 197–205.

[40] M. Abu Alsheikh, A. Selim, D. Niyato, L. Doyle, S. Lin, and H. Tan, "Deep activity recognition models with triaxial accelerometers," in *AAAI Conference on Artificial Intelligence*, vol. WS-16-01 - WS-16-15. United States: AI Access Foundation, 2016, pp. 8–13.

[41] K. Berggren, "Human activity recognition using deep learning and sensor fusion," Master's thesis, Lund University, 2018.

[42] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html

[43] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," 2015.

[44] S. A. Elkader, M. Barlow, and E. Lakshika, "Wearable sensors for recognizing individuals undertaking daily activities," in *Proceedings of the 2018 ACM International Symposium on Wearable Computers*, ser. ISWC

'18. New York, NY, USA: ACM, 2018, pp. 64–67. [Online]. Available: http://doi.acm.org/10.1145/3267242.3267245

[45] J. Mantyjarvi, M. Lindholm, E. Vildjiounaite, S. . Makela, and H. A. Ailisto, "Identifying users of portable devices from gait pattern with accelerometers," in *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 2, March 2005, pp. ii/973–ii/976 Vol. 2.

[46] V. R. Reddy, T. Chattopadhyay, K. Chakravarty, and A. Sinha, "Person identification from arbitrary position and posture using kinect," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '14. New York, NY, USA: ACM, 2014, pp. 350–351. [Online]. Available: http://doi.acm.org/10.1145/2668332.2668359

[47] J. Hernandez, D. J. McDuff, and R. W. Picard, "Bioinsights: Extracting personal data from "still" wearable motion sensors." in *BSN*. IEEE, 2015, pp. 1–6.

[48] G. Synnaeve and E. Dupoux, "A temporal coherence loss function for learning unsupervised acoustic embeddings," in *SLTU*, 2016.

[49] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4950–4954, 2015.

[50] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," vol. abs/1606.09549. Cham: Springer International Publishing, 2016.

[51] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *Arxiv*, 2016. [Online]. Available: https://arxiv.org/abs/1609.03499

[52] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," *CoRR*, vol. abs/1608.08242, 2016. [Online]. Available: http://arxiv.org/abs/1608.08242

[53] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," *CoRR*, vol. abs/1611.05267, 2016. [Online]. Available: http://arxiv.org/abs/1611.05267

[54] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 649–657. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969239.2969312

[55] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *2012 16th International Symposium on Wearable Computers*, June 2012, pp. 108–109.

[56] O. Banos, R. Garcia, J. A. Holgado-Terriza, M. Damas, H. Pomares, I. Rojas, A. Saez, and C. Villalonga, "mhealthdroid: A novel framework for agile development of mobile health applications," in *Ambient Assisted Living and Daily Activities*, L. Pecchia, L. L. Chen, C. Nugent, and J. Bravo, Eds. Cham: Springer International Publishing, 2014, pp. 91–98.

[57] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 2483–2493. [Online]. Available: http://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization.pdf

[58] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR*, vol. abs/1803.01271, 2018. [Online]. Available: http://arxiv.org/abs/1803.01271

[59] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2018.

[60] L. T. Nguyen, M. Zeng, P. Tague, and J. Zhang, "Recognizing new activities with limited training data," in *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, ser. ISWC '15. New York, NY, USA: ACM, 2015, pp. 67–74. [Online]. Available: http://doi.acm.org/10.1145/2802083.2808388

[61] S. Ha, J. Yun, and S. Choi, "Multi-modal convolutional neural networks for activity recognition," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, Oct 2015, pp. 3017–3022.

[62] S. Ha and S. Choi, "Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors," in *2016 International Joint Conference on Neural Networks (IJCNN)*, July 2016, pp. 381–388.

[63] D. H. Ballard, "Modular learning in neural networks," in *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, ser. AAAI'87. AAAI Press, 1987, pp. 279–284. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863696.1863746

[64] T. Plötz, N. Y. Hammerla, and P. Olivier, "Feature learning for activity recognition in ubiquitous computing," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, ser. IJCAI'11. AAAI Press, 2011, pp. 1729–1734. [Online]. Available: http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-290

[65] Y. Li, D. Shi, B. Ding, and D. Liu, "Unsupervised feature learning for human activity recognition using smartphone sensors," in *Mining Intelligence and Knowledge Exploration*, R. Prasath, P. O'Reilly, and T. Kathirvalavakumar, Eds. Cham: Springer International Publishing, 2014, pp. 99–107.

[66] L. Bai, C. Yeung, C. Efstratiou, and M. Chikomo, "Motion2vector: Unsupervised learning in human activity recognition using wrist-sensing data," in *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, ser. UbiComp/ISWC '19 Adjunct. New York, NY, USA: ACM, 2019, pp. 537–542. [Online]. Available: http://doi.acm.org/10.1145/3341162.3349335

[67] Y. Lu, Y. Wei, L. Liu, J. Zhong, L. Sun, and Y. Liu, "Towards unsupervised physical activity recognition using smartphone accelerometers," *Multimedia Tools and Applications*, vol. 76, no. 8, pp. 10 701–10 719, Apr 2017. [Online]. Available: https://doi.org/10.1007/s11042-015-3188-y

[68] Y. Kwon, K. Kang, and C. Bae, "Unsupervised learning for human activity recognition using smartphone sensors," *Expert Syst. Appl.*, vol. 41, pp. 6067–6074, 2014.

[69] E. Aljalbout, V. Golkov, Y. Siddiqui, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," *CoRR*, vol. abs/1801.07648, 2018. [Online]. Available: http://arxiv.org/abs/1801.07648

[70] O. Baños, M. Damas, H. Pomares, I. Rojas, M. A. Tóth, and O. Amft, "A benchmark dataset to evaluate sensor displacement in activity recognition," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12. New York, NY, USA: ACM, 2012, pp. 1026–1035. [Online]. Available: http://doi.acm.org/10.1145/2370216.2370437

[71] A. Reiss and D. Stricker, "Creating and benchmarking a new dataset for physical activity monitoring," in *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '12. New York, NY, USA: ACM, 2012, pp. 40:1–40:8. [Online]. Available: http://doi.acm.org/10.1145/2413097.2413148

[72] H.-T. Cheng, F.-T. Sun, M. L. Griss, P. Davis, J. Li, and D. You, "Nuactiv: recognizing unseen new activities using semantic attribute-based learning." in *MobiSys*. ACM, 2013, pp. 361–374.

[73] L. T. Nguyen, M. Zeng, P. Tague, and J. Zhang, "Recognizing new activities with limited training data," 09 2015.

[74] H.-T. Cheng, M. Griss, P. Davis, J. Li, and D. You, "Towards zero-shot learning for human activity recognition using semantic attribute sequence model," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 355?358. [Online]. Available: https://doi.org/10.1145/2493432.2493511

BIOGRAPHICAL STATEMENT

Taoran Sheng was born in Guiyang, Guizhou, China. He received his Ph.D. degree in Computer Science from the University of Texas at Arlington. Prior to the Ph.D. program in UTA, He received his M.S. degree in Computer Science from the University of Hamburg, Germany, and his B.Eng. degree in Telecommunications Engineering from Beijing University of Chemical Technology, China. His main research interests are deep learning, machine learning, human activity analysis and wearable computing.