

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

**SENSOR-BASED HUMAN ACTIVITY RECOGNITION VIA
ZERO-SHOT LEARNING**

WEI WANG
Interdisciplinary Graduate School
Joint NTU-UBC Research Centre of Excellence
in Active Living for the Elderly (LILY)

SENSOR-BASED HUMAN ACTIVITY RECOGNITION VIA ZERO-SHOT LEARNING

WEI WANG

**Interdisciplinary Graduate School
Joint NTU-UBC Research Centre of Excellence
in Active Living for the Elderly (LILY)**

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

25/Jan/2019

.....
Date

Wang Wei

.....
WEI WANG

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

25/3am/2019
.....
Date


.....
PROF. CHUNYAN MIAO

Authorship Attribution Statement

This thesis contains material from 4 paper(s) published in the following peer-reviewed journal(s) where I was the first and/or corresponding author.

Chapter 2 is published as Wei Wang, Vincent W. Zheng, Han Yu, and Chunyan Miao. A Survey of Zero-Shot Learning: Settings, Methods, and Applications. *ACM Transactions on Intelligent Systems and Technology (ACM TIST)*, 10(2):13:1-13:37, 2019. DOI: 10.1145/3293318.

The contributions of the co-authors are as follows:

- I reviewed the related literature, proposed the way to categorize and summarize the related literature, and wrote the paper.
- Dr. Vincent W. Zheng helped modify the organization of the paper, gave suggestions on the paper writing, and revised the paper.
- Prof. Han Yu gave suggestions on the content and organization of the paper, and revised the paper.
- Prof. Chunyan Miao provided guidance on the paper organization and the content.

Chapter 3 is published as Wei Wang, Chunyan Miao, and Shuji Hao. Zero-Shot Human Activity Recognition via Nonlinear Compatibility Based Method. *International Conference on Web Intelligence (WI 2017)*, ACM, pages 322-330, Leipzig, Germany, August 23-26, 2017. DOI: 10.1145/3106426.3106526.

The contributions of the co-authors are as follows:

- I proposed the method, designed and performed the experiments, and wrote the paper.
- Prof. Chunyan Miao supervised the study, guided the method proposal and the experiment design.
- Dr. Shuji Hao gave suggestions on the paper writing and experiment design, and revised the paper.

Chapter 4 is published as Wei Wang and Chunyan Miao. Activity Recognition in New Smart Home Environments. *ACM International Conference on Multimedia Workshop: International Workshop on Multimedia for Personal Health and Health Care (HealthMedia 2018)*, ACM, pages 29-37, Seoul, Republic of Korea, October 22, 2018. DOI: 10.1145/3264996.3265001.

The contributions of the co-authors are as follows:

- I proposed the method, designed and performed the experiments, and wrote the paper.
- Prof. Chunyan Miao provided guidance and revised the paper.

Chapter 5 is published as Wei Wang and Chunyan Miao. Multi-Resident Activity Recognition with Unseen Classes in Smart Homes. *IEEE International Conference on Ubiquitous Intelligence and Computing (UIC 2018)*, IEEE, pages 780-787, Guangzhou, China, October 8-12, 2018. DOI: 10.1109/SmartWorld.2018.00147.

The contributions of the co-authors are as follows:

- I proposed the method, designed and performed the experiments, and wrote the paper.
- Prof. Chunyan Miao supervised the study and revised the paper.

25/ Jan/ 2019

.....
Date

Wang Wei

.....
WEI WANG

Acknowledgments

I would like to express my sincere gratitude to my supervisor Prof. Chunyan Miao, for her patient guidance and constant encouragement. Over the past years, she gives me great help in the academic research. I also appreciate my co-supervisors Prof. Sinno Jialin Pan and Prof. Cyril Leung, and mentor Dr. Zhiqi Shen. They give a lot of help in my academic studies. I would appreciate Dr. Vincent Wenzhen Zheng and Prof. Han Yu. They give me effective discussions and suggestions for my research.

I express my thanks to my colleagues and friends, Yi Dong, Shuji Hao, Peng Cheng, Yong Liu, Siyuan Liu, Chi Zhang, Di Wang, Hao Zhang, Xuejiao Zhao, Huiguo Zhang, Rong Wang, and so on. They give me a lot of support and encouragement.

I would like to thank my parents. They give me continuous encouragement and love all the time.

In the published paper “Wei Wang, Han Yu, and Chunyan Miao. Deep Model for Dropout Prediction in MOOCs. *International Conference on Crowd Science and Engineering (ICCSE 2017)*, ACM, pages 26-32, Beijing, China, July 6-9, 2017. DOI: 10.1145/3126973.3126990”, the contributions of the co-authors are as follows: I proposed the method, designed and performed the experiments, and wrote the paper. Prof. Han Yu gave suggestions on the experiment design, and revised the paper. Prof. Chunyan Miao supervised the study, and provided guidance.

Contents

Acknowledgments	i
List of Figures	vi
List of Tables	vii
Abstract	viii
1 Introduction	1
1.1 Background and Motivations	1
1.1.1 Sensors as Data Collecting Devices	1
1.1.2 General Process of Sensor-Based Activity Recognition	2
1.1.3 Activity Recognition Including New Classes	4
1.1.4 Scope of This Thesis	6
1.1.5 Zero-Shot Learning	7
1.2 Problems and Methodologies	9
1.3 Summary of Contributions	15
1.4 Thesis Organization	16
2 Literature Review	17
2.1 Sensor-Based Activity Recognition	17
2.1.1 Sensors and Recognition Methods	18
2.1.2 Activity Recognition Including New Classes	20
2.1.3 Activity Recognition across Different Smart Homes	25
2.1.4 Multi-Resident Activity Recognition in Smart Homes	27
2.2 Zero-Shot Learning	29

2.2.1	Learning Settings	29
2.2.2	Semantic Spaces	30
2.2.3	Methods	35
2.2.4	Zero-Shot Learning in ARKN Problems	41
3	Unseen Activity Recognition for One Person	43
3.1	Background	43
3.1.1	Problem Description	44
3.1.2	General Process and Existing Works	46
3.2	Proposed Method	47
3.2.1	Data Preprocessing	47
3.2.2	Problem Formulation	48
3.2.3	Compatibility Based Methods	49
3.2.4	Proposed NCBM Method	51
3.3	Experiments	54
3.3.1	Datasets	54
3.3.2	Experimental Setup	55
3.3.3	Evaluation on Performance	61
3.3.4	Impact of Hidden Space Dimension	63
3.3.5	Evaluation on Model Learning Time	64
3.3.6	Discussion	66
3.4	Summary	67
4	Activity Recognition across Smart Homes with Different Binary Sen- sors and Label Spaces	68
4.1	Background	69
4.1.1	Binary Sensors	69
4.1.2	Activity Recognition across Different Smart Homes	70
4.2	Proposed Method	73

4.2.1	Problem Statement	73
4.2.2	Overview of the BSST Method	73
4.2.3	Feature Extraction	75
4.2.4	Prototype Defining	80
4.2.5	Compatibility Function	81
4.2.6	Discussion	83
4.3	Experiments	84
4.3.1	Datasets	84
4.3.2	Experimental Setup	85
4.3.3	Performance Evaluation	88
4.3.4	Evaluation on the Effect of Amount of Training Data	90
4.3.5	Discussion	92
4.4	Summary	93
5	Multi-Resident Activity Recognition Including Unseen Classes in Smart Homes	95
5.1	Background	96
5.1.1	Multi-Resident Activity Recognition in Smart Homes	96
5.1.2	Problem and Existing Works	97
5.2	Proposed Method	98
5.2.1	Problem Statement	99
5.2.2	Data Preprocessing	99
5.2.3	Proposed MRUA Method	100
5.2.4	Discussion	104
5.3	Experiments	105
5.3.1	Datasets	105
5.3.2	Experimental Setup	106
5.3.3	Performance Evaluation	108
5.3.4	Results on Each Testing Day	114

5.3.5	Effect of Amount of Training Data	114
5.3.6	Discussion	116
5.4	Summary	117
6	Conclusions and Future Work	119
6.1	Conclusions	119
6.2	Future Directions	120
A	Class Splitting, Defined Attributes and Meta-Features	123
A.1	Splitting of Classes, Defined Attributes and Prototypes in Chapter 3 . . .	123
A.2	Defined Meta-Features, Meta-Feature Vectors, Attributes, and Prototypes in Chapter 4	123
B	List of Publications	130
	References	131

List of Figures

1.1	General process of sensor-based activity recognition.	3
1.2	Activity recognition including new classes.	5
1.3	Research problems in this thesis.	13
2.1	Different semantic spaces in zero-shot learning.	30
2.2	Different categories of methods in zero-shot learning.	35
3.1	General process of methods for zero-shot learning problems under CIII learning setting.	46
3.2	Process of the NCBM method.	52
3.3	Performance of the NCBM method on the three datasets with different hidden space dimensions.	64
4.1	Process of the BSST method.	74
4.2	Feature extraction for a segment.	75
4.3	Effect of amount of training data.	91
5.1	Process of the MRUA method.	101
5.2	Results on each testing day.	115
5.3	Effect of amount of training data.	116

List of Tables

1.1	Characteristics of research problems in this thesis.	14
2.1	Some representative zero-shot learning methods belonging to different categories and under different learning settings.	40
3.1	Statistics of the number of instances and classes belonging to the seen and the unseen classes in each fold in the three datasets.	61
3.2	Comparison on average per-class accuracy (in %) of different methods. .	62
3.3	Comparison on model learning time (in seconds) of different methods. .	65
4.1	Transfer between OrdonezA and OrdonezB (in %).	89
5.1	Number of days each activity occurs.	108
5.2	Comparison of different methods (in %).	112
A.1	Class splitting in dataset PAMAP2.	124
A.2	Class splitting in dataset OPP.	124
A.3	Class splitting in dataset TUD.	124
A.4	Defined attributes and prototypes for dataset PAMAP2.	125
A.4	Defined attributes and prototypes for dataset PAMAP2 (continued). . . .	126
A.5	Defined attributes and prototypes for dataset OPP.	127
A.6	Defined meta-features and meta-feature vectors for datasets OrdonezA and OrdonezB.	128
A.7	Defined attributes and prototypes for datasets OrdonezA and OrdonezB.	129

Abstract

Sensor-based activity recognition aims to recognize human activities from the sensor readings. In recent years, with the advance of the sensor technology and the development of the machine learning algorithms, it obtains much progress. Activity recognition is a classification problem. Thus, existing works on sensor-based activity recognition are generally based on supervised-classification algorithms, such as SVM, HMM, CRF, kNN, and decision tree. In these works, for each class to recognize, sufficient labeled training instances belonging to this class are needed.

In practical applications, the process of obtaining labeled sensor readings is time consuming and costly. The activity classes having sufficient sensor readings are generally classes predefined during data collecting-and-labeling in the model learning phase. However, in many applications, the activities needed to recognize could contain not only the predefined classes, but also some previously unseen classes which do not have labeled training instances during model learning. In this thesis, we focus on sensor-based activity recognition including the unseen classes. Under this problem setting, the training instances belong to a set of activity classes (referred to as the seen classes), while the testing instances can come from the previously unseen activity classes (referred to as the unseen classes). For problems under this problem setting, as there are no labeled training instances belonging to the unseen classes, the zero-shot learning methods are used. We focus on three problems under this setting.

Firstly, we study the problem of unseen activity recognition for one person. In this problem, the classes covered by the training and the testing instances are disjoint. We propose a method, referred to as Nonlinear Compatibility Based Method (NCBM), to

address this problem. In our method, we adopt the binary attribute space as the semantic space, and use it to involve the semantic information. Then, we learn a compatibility function between the feature space and the semantic space. The recognition of the testing instances is achieved with the learned compatibility function and the unseen-class-prototypes in the semantic space. We conduct extensive experiments, and the experimental results show the effectiveness of our method.

Secondly, we study the problem of activity recognition across smart homes with different binary sensors and label spaces. We propose a method, referred to as Binary Sensor Semantic and Time Information Method (BSST), to address this problem. In this problem, the deployed binary sensor networks and the activities needed to recognize in different smart homes are different. In our method, we utilize the characteristics of the binary sensors, the semantic information about the sensors and the activities, and the time information. Sensor readings in different smart homes are extracted into a common feature space, and each activity class has a defined prototype in the semantic space. Then, we learn a classification model in one smart home environment and use it to perform activity recognition in a new smart home environment. During model learning, no information about the new smart home environment is needed. The experimental results show the effectiveness of our method.

Lastly, we study the problem of multi-resident activity recognition including unseen classes in smart homes. We propose a method, referred to as Multi-Resident with Unseen-Activity-Class Recognition Method (MRUA), to address this problem. In our method, the activity recognition for each resident is regarded as a learning task, and all tasks are learned jointly with multi-task learning techniques. The zero-shot learning techniques are also adopted. Thus, the unseen classes can be recognized. Through extensive experiments, we show the effectiveness of our method.

Chapter 1

Introduction

1.1 Background and Motivations

In many applications in the areas of ubiquitous computing and artificial intelligence, information about human activities is important. For example, in applications in smart homes [11], information about human activities is the basis of many context-aware services, such as behavioral monitoring, emergency detection and assisted living [4]. In another example, in security surveillance related applications [158], information about human activities is also needed. Other applications, such as energy saving offices [110] and context-based personal assistant [125], being aware of human activities is also necessary. In these applications, as the information about human activities is needed, how to recognize human activities in an automatic manner is a problem that is interesting and important. This problem is generally referred to as the problem of *human activity recognition*, and has obtained increasing attention over the past years [29, 38, 91].

1.1.1 Sensors as Data Collecting Devices

To automatically recognize human activities, devices that can collect information about activities are needed. In the past decade, with the advance of the sensor technology, various kinds of low-cost, low-power and miniaturized sensors have been utilized in our daily life [29, 38]. These sensors include the wearable sensors worn on the body of the

humans [91], and the ambient sensors deployed in the home or working environments [89]. Massive amount of sensor readings are generated by them, and these sensor readings contain information about human activities [29, 38]. In view of this, sensors can be utilized as activity-information collecting devices.

Apart from the sensors, another kind of widely used activity-information collecting device are the cameras, such as the surveillance cameras. Through the cameras, a wealth of information about the human activities is recorded in the generated videos [29]. In the area of computer vision, there have been large amount of research works on video-based human activity recognition [80, 121, 150]. However, compared with the cameras, the sensors have many advantages. Firstly, they are more privacy preserving and less intrusive [146, 155]. Cameras are generally considered as recording devices [29], and often regarded to be intrusive [16]. Furthermore, in many application scenarios, such as applications in the home environment, the privacy is important [155]. In these applications, it is not suitable to use the cameras. Secondly, as the cameras are often considered as recording devices. When using them to collect information about the activities, the humans may behave unnaturally [15]. This can influence the quality of the recorded information. On the contrast, this problem is less severe when using sensors as information collecting devices. Thirdly, in view of computational complexity [32], the videos generated by the cameras are more computationally expensive than the sensor readings. The videos are generally a type of data with high dimension, and have expensive computation cost. On the contrast, the cost of dealing with the sensor readings is relatively low. With the above advantages, sensors have got increasingly wide adoption in human activity recognition applications [16, 24, 29, 38, 91].

1.1.2 General Process of Sensor-Based Activity Recognition

Adopting sensors as data-collecting devices, we can get the sensor readings. The sensor readings contain information about human activities, and activity recognition is per-

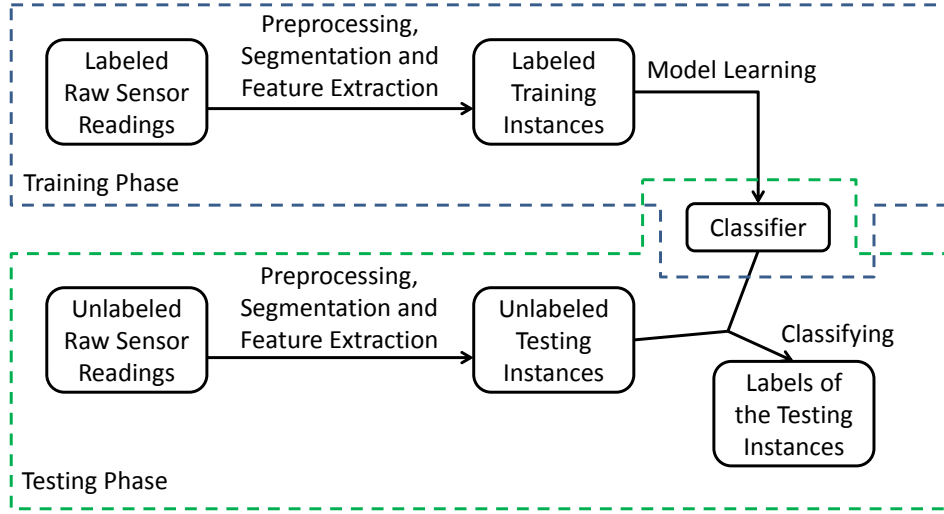


Figure 1.1: General process of sensor-based activity recognition.

formed based on them. In recent years, the area of machine learning has achieved significant progress, and machine learning methods have been widely used in sensor-data related problems [38]. The goal of activity recognition is to predict the activity class labels. It is a classification problem. Thus, existing works on sensor-based activity recognition generally address this problem through supervised classification algorithms, such as SVM [51], CRF [89], HMM [155], and decision tree [18]. As shown in Figure 1.1, the general process of sensor-based activity recognition is as follows [24, 38]:

- *Training Phase:* In the training phase, firstly some sensor readings belonging to a set of activity classes are collected and labeled. Then in the preprocessing stage, some data preprocessing operations are performed, such as synchronizing the readings from different sensors and removing the artifacts in the data [24]. After that is the segmentation stage. The raw sensor readings are time series data. They are generally segmented into a set of segments. In the feature extraction stage, for each segment, some features (such as statistical features like mean value, standard deviation, and spectral features like spectral centroid, spectral entropy [38]) are extracted from the sensor readings. For each segment, generally one labeled instance in the feature space is extracted. With

the extracted labeled training instances, through some machine learning method (the supervised classification algorithm), the classifier for the instances is learned.

- *Testing Phase*: In the testing phase, being similar to the training phase, some sensor readings are collected from the sensors. Also through the stages of preprocessing, segmentation and feature extraction, the testing instances in the feature space are extracted. With the classifier learned in the training phase, class labels of the testing instances are predicted.

1.1.3 Activity Recognition Including New Classes

In practical applications, the above general process adopted in many existing methods has several limitations. One limitation is that, in these methods, for each class to recognize, sufficient labeled training instances belonging to this class are needed [109]. In many practical applications, the classes having sufficient training instances are generally activity classes predefined when collecting and labeling sensor readings in the training phase. However, in many applications, the predefined classes may not cover all of the activity classes needed to recognize. The classes needed to recognize could include some new activity classes. This is because, on the one hand, *the number of activities humans can perform is large* [10, 34]; on the other hand, *in many practical applications, if the application scenario has changed, the set of activities needed to recognize can also change* [2]. Here, we give the definitions of the *existing classes* and the *new classes*.

Definition 1.1 (Existing Classes). The activity classes that are predefined during sensor reading collecting-and-labeling in the training phase are referred to as the existing classes.

Definition 1.2 (New Classes). The activity classes that are not predefined but also needed to recognize in the testing phase are referred to as the new classes.

To deal with the problem of activity recognition including new classes, a straightforward approach is to collect and label sufficient sensor readings for the new classes, and

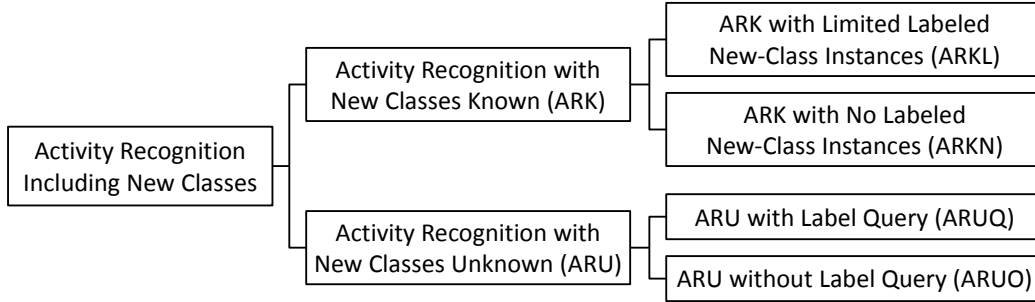


Figure 1.2: Activity recognition including new classes.

retrain a new classifier with the newly collected data. However, in practical applications, the process of obtaining sufficient labeled training instances is time consuming and costly [10, 34, 172]. Furthermore, in some applications, the new classes needed to recognize may be determined in the testing phase, and the activity recognition needs to be performed in an online manner. In such situations, it is not practical to obtain sufficient sensor readings for the new classes and retrain a new classifier. Methods that can deal with the new classes are needed. On account of this, in recent years, there have been increasing amount of research works on sensor-based activity recognition including the new classes.

Different problem settings. In existing works on activity recognition including new classes, the recognition problems under different problem settings have been studied. For example, in some problems, when the application scenario has changed, the set of activities needed to recognize also changes. The new classes that are needed to recognize have limited amount of training instances [82, 109]. In some problems, the new classes do not have labeled training instances [34, 180]. In addition, in some problems, we do not know the labels of the new classes at the beginning, and need to discover them [39]. We organize the different problem settings covered by existing works as shown in Figure 1.2. A detailed review of works on activity recognition including new classes will be given in Section 2.1.2.

As shown in this figure, in some problems, the new classes needed to recognize are

determined, i.e., we have decided the new classes that are needed to recognize. Thus, the labels of the new classes are known. We refer to this problem setting as *activity recognition with new classes known (ARK)*. In other problems, the new classes that are needed to recognize are not determined, and the labels of the new activities are unknown. The new classes need to be discovered. We refer to this problem setting as *activity recognition with new classes unknown (ARU)*.

Under the setting of ARK, in some problems, there are limited amount of labeled new-class instances in the training data. We refer to this problem setting as *ARK with limited labeled new-class instances (ARKL)*. While in other problems, there are no labeled instances belonging to the new classes in the training data, and we refer to this problem setting as *ARK with no labeled new-class instances (ARKN)*.

Under the setting of ARU, the new classes need to be discovered. In some problems, for the new discovered activity classes, there exists a process of querying their class labels. We refer to this problem setting as *ARU with label query (ARUQ)*. While in other problems, the activity labels of the discovered new classes are not queried. We refer to this problem setting as *ARU without label query (ARUO)*.

1.1.4 Scope of This Thesis

In this thesis, we focus on the problems under the setting of ARKN. Under this problem setting, the training data just contain labeled instances belonging to the existing classes, and no labeled instances belonging to the new classes are available. However, the testing instances needed to classify can come from the new classes.

Knowledge transfer. Under the problem setting of ARKN, we need the knowledge contained in the training instances to be transferred to the task of testing-instance recognition. The activity classes covered by the training and the testing instances are different. Thus, transfer learning [115] techniques are generally needed.

Transfer learning is a learning paradigm in which the knowledge contained in the source domain and source task is transferred to the target domain for learning the model in the target task [115]. Under the problem setting of ARKN, we need the knowledge transfer across classes, i.e., from the existing classes to the new classes. In existing works on transfer learning across classes, for each class to recognize, some labeled instances belonging to this class are needed. However, under the setting of ARKN, no labeled instances belonging to the new classes are available. Thus, these transfer learning approaches generally cannot be used. As a subarea of transfer learning, *zero-shot learning* techniques can be utilized for problems under the ARKN setting.

1.1.5 Zero-Shot Learning

Zero-shot learning [166] is a learning paradigm which aims to classify instances belonging to classes that have no labeled instances. In zero-shot learning, there are some labeled training instances in the feature space. The classes covered by these training instances are referred to as the *seen classes*. In the feature space, there are also some unlabeled testing instances, which belong to another set of classes. These classes are referred to as the *unseen classes*. The feature space is usually a real number space and each instance is represented as a vector within it. Each instance is usually assumed to belong to one class. Now, we give the definition of zero-shot learning. Denote $\mathcal{S} = \{c_i^s | i = 1, \dots, N_s\}$ as the set of seen classes, where each c_i^s is a seen class. Denote $\mathcal{U} = \{c_i^u | i = 1, \dots, N_u\}$ as the set of unseen classes, where each c_i^u is an unseen class. Note that $\mathcal{S} \cap \mathcal{U} = \emptyset$. Denote $\mathcal{X}$ as the feature space, usually it is a real number space. Denote $D^{tr} = \{(\mathbf{x}_i^{tr}, y_i^{tr}) \in \mathcal{X} \times \mathcal{S}\}_{i=1}^{N_{tr}}$ as the set of labeled training instances belonging to seen classes, for each labeled instance $(\mathbf{x}_i^{tr}, y_i^{tr})$, $\mathbf{x}_i^{tr}$ is the instance in the feature space, and y_i^{tr} is the corresponding class label. Denote $X^{te} = \{\mathbf{x}_i^{te} \in \mathcal{X}\}_{i=1}^{N_{te}}$ as the set of testing instances, where each $\mathbf{x}_i^{te}$ is a testing instance in the feature space. Denote $Y^{te} = \{y_i^{te} \in \mathcal{U}\}_{i=1}^{N_{te}}$ as the corresponding class labels for X^{te} , which are to be predicted.

Definition 1.3 (Zero-Shot Learning). Given labeled training instances D^{tr} belonging to the seen classes $\mathcal{S}$, zero-shot learning aims to learn a classifier $f^u(\cdot) : \mathcal{X} \rightarrow \mathcal{U}$ that can classify testing instances X^{te} (i.e., to predict Y^{te}) belonging to the unseen classes $\mathcal{U}$.

Auxiliary information. As no labeled instances belonging to the unseen classes are available, to solve the zero-shot learning problem, some *auxiliary information* is necessary. Such auxiliary information should contain information about all of the unseen classes. This is to guarantee each of the unseen classes is provided with corresponding auxiliary information. Meanwhile, the auxiliary information should be related to the instances in the feature space. This is to guarantee the auxiliary information is usable.

In existing works, the approach to involve auxiliary information is inspired by the way human beings recognize the world. Humans can perform zero-shot learning with the help of some semantic background knowledge. For example, with the knowledge that “the activity jogging is similar to the activity running, but with a slower speed”, we can recognize the activity “jogging” even without having seen one performing it before, as long as we know what the activity “running” is like and what is “slower speed”. In this way, the auxiliary information involved by existing zero-shot learning methods is usually some *semantic information*. It forms a space which contains both the seen and the unseen classes. As this space contains semantic information, it is often referred to as the *semantic space*. Being similar to the feature space, the semantic space is also usually a real number space. In the semantic space, each class has a corresponding vector representation, which is referred to as the *class prototype* (or *prototype* for short) of this class. We denote $\mathcal{T}$ as the semantic space. Denote $\mathbf{t}_i^s \in \mathcal{T}$ as the class prototype for seen class c_i^s , and $\mathbf{t}_i^u \in \mathcal{T}$ as the class prototype for unseen class c_i^u . Denote $T^s = \{\mathbf{t}_i^s\}_{i=1}^{N_s}$ as the set of prototypes for seen classes, and $T^u = \{\mathbf{t}_i^u\}_{i=1}^{N_u}$ as the set of prototypes for unseen classes. In zero-shot learning, along with the training instances D^{tr} , the class prototypes T^s and T^u are also involved in obtaining the zero-shot classifier $f^u(\cdot)$.

In existing works on zero-shot learning, various semantic spaces have been involved, and different zero-shot learning methods have been proposed. In Section 2.2, we will give a review of existing works on zero-shot learning.

1.2 Problems and Methodologies

In this thesis, we focus on the sensor-based activity recognition problems under the setting of ARKN. As stated above, under this problem setting, no labeled instances belonging to the new classes are available, and generally zero-shot learning methods are adopted. Under this problem setting, the existing classes accord with the definition of the seen classes in zero-shot learning; the new classes accord with the definition of the unseen classes. Following some existing works [33, 34] under this problem setting, in the following, when we introduce our research works under the ARKN setting, we use the term “seen classes” instead of “existing classes”, and “unseen classes” instead of “new classes”. The ARKN problem setting can also be referred to as “sensor-based activity recognition including unseen classes”.

We study three research problems: (1) unseen activity recognition for one person; (2) activity recognition across smart homes with different binary sensors and label spaces; and (3) multi-resident activity recognition including unseen classes in smart homes. Next, we present these problems and introduce the proposed methods.

Unseen activity recognition for one person. In sensor-based activity recognition problems, in many applications, instances belonging to the unseen classes can be encountered. Furthermore, in some applications, the activities covered by the training data and the activities needed to recognize are disjoint. For example, in some applications, the training instances are collected in one scenario (such as in the working environment), while the activity recognition needs to be performed in another scenario (such as in the home environment). The activities covered by them are disjoint.

Problem. We study the problem in which, for one person, the activities covered by the training and the testing instances are disjoint, i.e., the activities needed to recognize belonging to the unseen classes. It is a zero-shot learning problem.

Challenges. The set of activity classes covered by the training and the testing data are disjoint. How to effectively use the training data and the auxiliary information to learn the classification model for the testing instances is the main challenge.

Existing works. Existing methods for the problem of sensor-based zero-shot activity recognition have some limitations. In some methods [33, 34], the optimization in the training phase is for each dimension of the semantic space independently, not for the whole classification process. In some methods [180], the unseen classes to recognize need to be known in the training phase. The classifier is learned for a fixed set of unseen classes. If there are new unseen classes needed to recognize, they have to retrain a new model.

Proposed method. We propose a method, referred to as Nonlinear Compatibility Based Method (NCBM), for sensor-based zero-shot activity recognition. In this method, in the training phase, it learns a compatibility function between feature space instances and semantic space prototypes. In the testing phase, with the learned compatibility function, each testing instance is classified to the unseen class with the highest compatibility score. In our method, in the training phase, no information about the testing data is needed. This makes our method being suitable for many practical applications.

Activity recognition across smart homes with different binary sensors and label spaces. Smart homes are generally houses equipped with different devices to facilitate the living of the residents [11]. In smart homes, many context-aware services can be provided, and these services need the information about the residents' activities. To collect data about the activities, generally ambient sensors deployed in the houses are used. And in each smart home environment, the deployed ambient sensors form a

sensor network. Among different ambient sensors, binary sensors have got wide adoption. Binary sensors are sensors the output of which is binary values. Examples of the binary sensors are switch sensors attached on the doors and pressure sensors attached on the seats [38]. With the deployed sensors, activity recognition is performed based on the sensor readings.

Most of existing activity recognition methods are based on supervised classification algorithms. One drawback of these methods is that, the classifier learned in one smart home environment usually cannot be used in another environment. For a new smart home environment, sufficient labeled sensor readings from this environment have to be collected to learn the needed classifier. This process is time consuming and expensive. If labeled sensor readings collected in some existing smart home (following [154], we refer to it as the *source house*) can be utilized to learn the needed classifier in a new smart home (following [154], we refer to it as the *target house*), the cost of getting labeled instances will decrease to a large extent. In the target house, the sensor network and/or the class label space can be different from that in the source house.

Problem. We study the problem of activity recognition across smart homes with both the sensor networks and the label spaces being different. Particularly, we focus on the sensor networks consist of binary sensors.

Challenges. The sensor networks and the label spaces in the source and the target houses are different. How to utilize the sensor readings in the source house and the auxiliary information to learn the classification model for the target house is the main challenge.

Existing works. Existing works [74, 127] on problem of activity recognition across smart homes with different sensor networks and label spaces generally need the testing instances in the target house during the model learning phase, i.e., the methods are under transductive setting. This limits the application scope of these methods.

Proposed method. We propose a method, referred to as Binary Sensor Semantic and Time Information Method (BSST), for this problem. In this method, the sensor readings from different sensor networks are extracted into a common feature space, and the class labels in different label spaces are represented in a common semantic space. The classification model is learned with the data in the source house. Different from existing methods which need testing instances during model learning, in our method, during model learning, no information about the target house is needed. Thus, compared with many existing methods, our method is more suitable for practical applications.

Multi-resident activity recognition including unseen classes in smart homes.

As stated above, in smart homes, the ambient sensors deployed in the houses are generally used as data collecting devices. The majority of existing works on smart home activity recognition are under the setting of single resident, i.e., there is only one resident in the home environment [3, 112, 155]. All sensor readings are solely about this resident. However, in many scenarios, the residents do not live alone. In a home environment, there are usually multiple residents.

Compared with the single-resident setting, activity recognition under the multi-resident setting is more challenging. Under this setting, the sensor readings about different residents are mixed together. In addition, the activities of one resident may be influenced by the activities of other residents. Existing works on multi-resident activity recognition are usually based on supervised classification methods, and can only recognize the seen classes that have labeled instances in the training data. However, in many applications, the activities needed to recognize can contain not only the seen classes, but also the unseen classes.

Problem. We study the problem of multi-resident activity recognition including unseen classes in smart homes. In this problem, not only activities coming from multiple residents are needed to recognize, but also these activities contain the unseen activities.

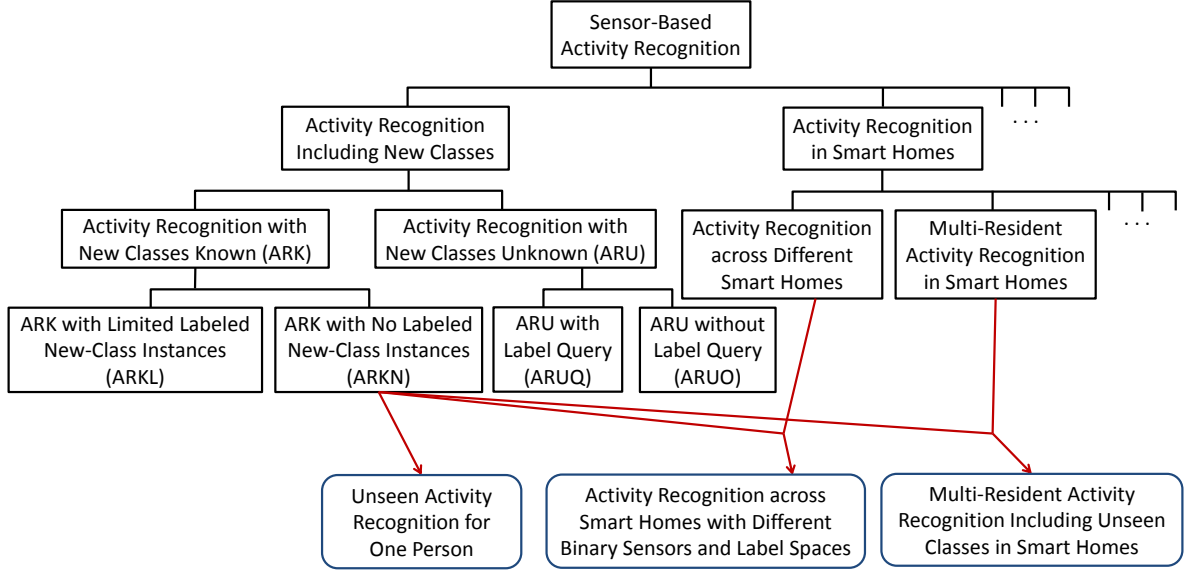


Figure 1.3: Research problems in this thesis.

Challenges. The activities of more than one resident are needed to recognize. In addition, the activities needed to recognize contain the unseen activities. How to use the training data and the auxiliary information to learn the recognition model that not only can recognize activities of multiple residents, but also can recognize the unseen activity classes is the main challenge.

Existing works. There are existing works on multi-resident activity recognition [19], but they are not for the case in which there are unseen classes. On the other hand, existing works on sensor-based unseen class recognition generally focus on the setting of one person [34, 180].

Proposed method. We propose a method, referred to as Multi-Resident with Unseen-Activity-Class Recognition Method (MRUA), to address this problem. Our MRUA method utilizes the techniques of multi-task learning and zero-shot learning. It regards the activity recognition of each resident as a learning task, and learns all tasks jointly. By utilizing zero-shot learning techniques, unseen classes can be recognized.

Table 1.1: Characteristics of research problems in this thesis.

Problem	Number of Persons	Sensors in the Training and the Testing Data
Unseen activity recognition for one person	One	Same
Activity recognition across smart homes with different binary sensors and label spaces	One	Different
Multi-resident activity recognition including unseen classes in smart homes	Two or more	Same

Summary. The three research problems stated above are shown in Figure 1.3. From the figure we can see, all of the three problems are under the setting of ARKN. The problem of “activity recognition across smart homes with different binary sensors and label spaces” can be seen as in the intersection of ARKN and “activity recognition across different smart homes”; while the problem of “multi-resident activity recognition including unseen classes in smart homes” can be seen as in the intersection of ARKN and “multi-resident activity recognition in smart homes”.

In all of the three problems, as there are instances belonging to the unseen classes needed to recognize, the zero-shot learning techniques are utilized. The characteristics of these problems are summarized in Table 1.1. In the problem “unseen activity recognition for one person”, activities of one person are needed to recognize. The sensor readings in the training and the testing data are from the same set of sensors. The problem “activity recognition across smart homes with different binary sensors and label spaces” is in the area of activity recognition in smart homes. In this problem, the activities of one resident are needed to recognize, but the sensors used in the training and the testing phase are different. The problem “multi-resident activity recognition including unseen classes in smart homes” is also in the area of activity recognition in smart homes. In this problem, the activities of more than one resident are needed to recognize, while the sensors used in the training and the testing phase are the same.

1.3 Summary of Contributions

In this thesis, we make contributions in the area of sensor-based activity recognition under ARKN setting. The main contributions are summarized as follows:

- We study the problem of unseen activity recognition for one person, which is a zero-shot learning problem. To address this problem, we propose a method NCBM. To the best of our knowledge, this is the first work utilizing compatibility function in problems of sensor-based zero-shot activity recognition. In our method, during model learning, the compatibility function is learned with the training instances and the seen class prototypes, no information about the testing data is needed. Classification of the testing instances is performed with the learned compatibility function. We conduct extensive experiments on three publicly available datasets. Experimental results show the effectiveness of our method.
- We study the problem of activity recognition across smart homes with different binary sensors and label spaces. We propose a method BSST to address this problem. To the best of our knowledge, this is the first work for activity recognition across smart homes with different sensor networks and label spaces, in which during model learning, no information about the new smart home environment is needed. Particularly, we focus on sensor networks consist of binary sensors. Our method utilizes the characteristics of binary sensors, the semantic information of the sensors and the activities, and the time information. To evaluate the effectiveness of our method, we conduct extensive experiments on publicly available real-world datasets.
- We study the problem of multi-resident activity recognition including unseen classes in smart homes. We propose a method MRUA to address this problem. To the best

of our knowledge, this is the first work focusing on the problem of multi-resident activity recognition including unseen classes in smart homes. Our method utilizes the techniques in multi-task learning and zero-shot learning. It regards the activity recognition of each resident as a learning task, and learns all tasks jointly. As zero-shot learning techniques are utilized, the unseen classes can be recognized. We conduct extensive experiments on publicly available real-world datasets, and show the effectiveness of our method.

1.4 Thesis Organization

This thesis is organized as follows. In Chapter 2, we review related works on sensor-based human activity recognition and zero-shot learning. In Chapter 3, we introduce the proposed NCBM method for unseen activity recognition for one person. In Chapter 4, we introduce the proposed BSST method for activity recognition across smart homes with different binary sensors and label spaces. In Chapter 5, we introduce the proposed MRUA method for multi-resident activity recognition including unseen classes in smart homes. Finally, in Chapter 6, we conclude this thesis and discuss some future research directions.

Chapter 2

Literature Review

As introduced in Chapter 1, in this thesis, we focus on the problems of sensor-based activity recognition under the setting of ARKN. To address the problems we focused on, zero-shot learning techniques are used. In Section 2.1, we review related works in the area of sensor-based activity recognition. In Section 2.2, we review related works on zero-shot learning.

2.1 Sensor-Based Activity Recognition

In Section 2.1.1, we review related works in the general area of sensor-based activity recognition, and we focus on two aspects: sensors and recognition methods. The problems we focused on in this thesis are in the area of ARKN. As shown in Figure 1.2, it is a subarea of sensor-based activity recognition including new classes. Thus, in Section 2.1.2, we focus on the related works in the area of sensor-based activity recognition including new classes. In this thesis, the work introduced in Chapter 4 is also in the area of “activity recognition across different smart homes”. Thus, in Section 2.1.3, we review related works in this area. The work introduced in Chapter 5 is also in the area of “multi-resident activity recognition in smart homes”. Thus, in Section 2.1.4, we review related works in this area.

2.1.1 Sensors and Recognition Methods

Sensor-based activity recognition has received extensive research in the past years [24, 29, 83]. In sensor-based activity recognition, the data come from the sensor readings, and various recognition methods are proposed. The sensors and the recognition methods are two important components of sensor-based activity recognition. In this section, we review related works on these two aspects.

2.1.1.1 Sensors

With the advance of the sensor techniques, various types of sensors have been utilized in activity recognition. The widely used sensors mainly include the *wearable sensors* and the *ambient sensors* [29, 38].

Wearable sensors are sensors worn on the body of the humans. In existing works, various kinds of wearable sensors have been used, such as the accelerometers, the gyroscopes, and the magnetometers [38]. They are embedded in different devices, such as the smart phones, and the sports bracelets. As these sensors are worn on the body of the humans, they can get information about the pose and the movement of the humans [29]. Thus, they are suitable for recognizing activities which can mainly be described by the human pose-and-movement information, like walking, running, and jumping. In addition, as they are worn on the body of the humans, they are not restricted to the environment, and can be carried by the humans to different environments. However, this characteristic also makes the wearable sensors just can capture little information about the environment. Thus, they are not suitable for recognizing activities which involve many interactions with the environment [29].

Ambient sensors are sensors deployed in the environment. They can be sensors attached on the objects, such as the pressure sensors attached on the seats, and the switch sensors attached on the cabinets [38]. Also they can be sensors deployed at some fixed

locations in the house, such as the passive infrared sensors attach on the wall or on the ceiling. These sensors usually form a sensor network in certain environment, such as the home environment or the working environment. As these sensors are deployed in the environment, information about the environment can be captured by them. Thus, they are suitable for recognizing activities which have interactions with the environment. In applications related with smart homes, the activities we aim to recognize are generally daily living activities like having breakfast, taking a shower, and watching TV. These activities involve large amount of interactions with the environment. Thus, in existing works on smart home activity recognition, ambient sensors are widely used [89, 146, 155]. Among different types of the ambient sensors, a large percentage of them belong to the binary sensors [112, 146, 155]. Binary sensors are sensors the out of which are binary values. They can be state-change sensors, like the switch sensors and the passive infrared sensors [38, 155]. For some other sensors, such as the temperature sensors or humidity sensors, the output of them can also be transformed into binary values by setting some threshold values [155]. The restriction of ambient sensors is that they are deployed in a certain environment, and can just be used for activity recognition within it.

2.1.1.2 Recognition Methods

Activity recognition is a classification problem. To get the class labels of the sensor readings, the majority of existing works are based on machine learning algorithms [38]. Thus, many supervised-classification algorithms in the area of machine learning are applied, such as SVM [51], kNN [120], CRF [89], HMM [155], decision tree [18], and Naive-Bayes classifier [101]. Especially in recent years, with the fast development of the deep learning techniques, many methods [70, 113, 171] based on deep learning are proposed.

However, for supervised-classification based methods, sufficient training instances are needed for each activity to recognize. In practical applications, the process of obtaining

labeled instances is expensive [10, 34, 172]. On account of this, recognition methods under other learning paradigms are proposed. Some methods [64, 144, 172] are under *semi-supervised learning* paradigm. In these methods, along with the labeled instances, the unlabeled instances are also utilized in model learning. In this way, the burden of instance labeling is relieved. Some methods [14, 144] are under *active learning* paradigm. In these methods, selected instances are queried for labels. The burden of instance labeling is also relieved. Some methods [69, 126] are under *transfer learning* paradigm. In these methods, information contained in instances in the source domain is transferred for activity recognition in the target domain.

Besides the methods based on machine learning algorithms, there also exist some logic-based [22] and ontology-based [134] methods. These methods do not need the large amount of training data as in machine learning based methods. However, in these methods, to get the logical formalisms or the ontology information which are necessary for activity recognition, large amount of domain knowledge is needed. This needs the domain experts to pay a lot of effort. There are also methods [118, 164] using the information got from the web texts for activity recognition. However, these methods are mainly suitable for activities that have clear steps and can be described by the interacted objects.

2.1.2 Activity Recognition Including New Classes

General activity recognition methods just focus on the existing classes which are pre-defined during the sensor reading collecting-and-labeling stage in the training phase. However, as introduced in Section 1.1.3, in practical applications, there could exist new classes needed to recognize. The different problem settings related with new class recognition covered by existing works are organized in Figure 1.2. In Section 1.1.3, we have given a brief introduction of different problem settings. In this section, we review related works under each problem setting.

2.1.2.1 Activity Recognition with New Classes Known (ARK)

Under the ARK problem setting, the labels of the new classes are known. According to whether there are labeled training instances for the new classes, the ARK setting can be classified into *ARK with limited labeled new-class instances (ARKL)* and *ARK with no labeled new-class instances (ARKN)*.

ARK with limited labeled new-class instances (ARKL). Under this problem setting, the labels of the new classes are known in the training phase. The existing classes have sufficient training instances, while the new classes just have limited amount of training instances.

Some problems are under the setting of transfer between two datasets. In these problems, in one dataset (referred to as the *source dataset*), there are sufficient labeled training instances belonging to the existing classes. In another dataset (referred to as the *target dataset*), the labeled training instances cover the existing and the new classes. However, the amount of instances in the target dataset is limited. Thus, the amount of training instances for the new classes is limited. Different methods are proposed for activity recognition in the target dataset, which uses the instances in the source dataset. In [82], an UnTran framework is proposed. In this framework, in the training phase, with the instances in the source dataset, a classifier for the existing classes is learned; with the instances in the target dataset, two classifiers for the existing and the new classes are learned. Meanwhile, with the instances in the source dataset, a one-class SVM is learned to determine the instance belonging to the existing classes or the new classes. In the testing phase, for a testing instance, with the learned one-class SVM, it is firstly determined belonging to the existing or the new classes. If it belongs to the existing classes, combination of the classification results of classifiers in the source dataset and the target dataset is used. If it belongs to the new classes, combination of the classification results of classifiers in the target dataset is used. In [81], a TransAct

framework is proposed. In this framework, with the instances from the source and the target datasets, a classifier to determine whether a testing instance belongs to the existing classes is learned. If the instance belongs to the existing classes, it is classified with the multi-class transfer boost classifier. If it belongs to the new classes, it is classified with the cluster-based classifier.

In some methods, the activities are regarded as composed of some components, and the classifiers for the basic components are learned. In [21], the focus is recognizing composite activities. The composite activities are assumed to be composed of the activity events. The goal of the method is using the information about the activity events to facilitate the recognition of the composite activities. With the training data in the source dataset, a layered model is learned using CRFs, and the classifiers for the activity events are contained in it. Then, with the learned classifiers for the activity events, in the target dataset (which contains new classes), fewer training instances are needed.

In some problems, in one dataset, there are sufficient training instances for the existing classes and limited training instances for the new classes. In [109], a method FE-AT is proposed. In this method, the semantic attributes of the activities are utilized. The semantic attributes are description terms of the activities. With the training instances, three classifiers are learned. The first is feature-based classifier, the second is attribute-based classifier, and another is a classifier to determine whether a testing instance belongs to the existing classes or the new classes. For a testing instance, it is firstly determined belonging to the existing classes or the new classes. If it belongs to the existing classes, classification results of feature-based classifier are applied; otherwise, classification results of feature-based classifier and attribute-based classifier are applied. In [10], the recognition of activities is performed by a hierarchical active transfer learning framework. In which transfer learning and active learning techniques are utilized.

ARK with no labeled new-class instances (ARKN). Under this problem setting, the new classes (unseen classes) have no labeled instances. To address problems under

this problem setting, zero-shot learning methods are generally needed. We will introduce existing works under this problem setting in Section 2.2.4 after introducing related works on zero-shot learning.

2.1.2.2 Activity Recognition with New Classes Unknown (ARU)

Under the ARU problem setting, being different from the problem setting of ARK, the labels of the new classes are unknown. Thus, a process of “activity discovery” is needed. Here, “activity discovery” refers to the process of discovering potential activities [38, 39, 174]. Through activity discovery, some potential activities are discovered. They can belong to the existing classes or the new classes. After the potential activities being discovered, we just know the existence of them, but we do not know their class labels. In some methods, there is a process of “querying the class labels”, which queries the labels of the discovered potential activities; while in other methods, there is no such a process. Based on this, we categorize the ARU problem setting into *ARU with label query (ARUQ)* and *ARU without label query (ARUO)*.

ARU with label query (ARUQ). Under this problem setting, the class labels of the new classes are unknown at the beginning. Firstly, there is a process of “activity discovery”. In which, the labels of the discovered potential activities are queried. After the process of label querying, the labels of the new classes are known. Under this problem setting, as label query is required, active learning are involved in the recognition methods.

In some problems, the labeled training instances come from the existing classes; meanwhile, there are some unlabeled training instances, which come from the new classes. In [72], clustering algorithms are firstly applied to the unlabeled instances, and some instance is selected in each cluster (this is the process of activity discovery). The selected instances are queried for labels (this is the process of query label for discovered activities). After the label query, the class labels of the newly discovered activities (which can

belong to the existing classes or the new classes) get known, and some labeled instances belonging to the new classes are obtained. With the labeled instances, the classifier is learned.

In some methods, the labeled training instances come from the existing classes, and the classifier for the existing classes is learned with them. In the testing phase, potential activities (can belong to the existing classes or the new classes) are discovered and queried for labels, and the classifier is adapted. In [174], for each existing class, with the labeled training instances belonging to this class, some statistics parameters for it are calculated. The testing instances are classified to the existing classes with these statistics parameters, and these statistics parameters are updated with the classified testing instances. There are also some testing instances not classified (this is the process of activity discovery), and they form a candidate pool. Then, label query is performed for selected instances (this is the process of label query). After label query, labels of the new classes can be known, and labeled instances belonging to the new classes can be obtained. In method SuperAD [107], with the labeled training instances, a classifier for the existing classes is learned. In the testing phase, with the learned classifier, the testing instances are classified into the existing classes. For each existing class, a Gaussian Mixture Model (GMM) is learned with the testing instances classified into this class. With the learned GMMs, each testing instance is predicted the probabilities of belonging to each existing class. Testing instance with the lowest probability is selected to query for label (process of activity discovery and label query). Then the classifier is retrained.

In some methods [1], the classifier is firstly learned with the training instances for the existing classes. Then, in the testing phase, the classifier is adapted with not only adding the newly discovered new classes, but also abandoning the outdated classes. This is performed by active learning.

ARU without label query (ARUO). Under this problem setting, there is no process of label query for newly discovered potential activities. Thus, being a little different from

the definition of new classes in Definition 1.2, under this problem setting, the class labels of the new classes do not need to be recognized.

In [39], there are many training instances not belonging to any existing classes. These instances contain the new classes. Through clustering algorithms, activity patterns (the potential activities) are found from these instances (this is the process of activity discovery). However, as there is no process of label querying, the labels of the discovered activity patterns are unknown. These activity patterns are used along with the existing activity classes during the classifier learning. With the participation of these discovered activity patterns, the performance of the learned classifier for the existing classes is improved. In [128], there is also a process of activity pattern discovery. The discovered activity patterns are used for activity tracking. From the above introduction we can see, for problems under this problem setting, we do not aim to recognize the new classes. There is just a process of activity discovery. The discovered activity patterns/potential activities can contain new classes. This can facilitate other learning problems, such as classification of the existing classes or activity tracking.

In some methods [108] related with unknown new classes, there is no process of activity discovery. When learning the classifier for the existing classes, they consider the existence of these unknown new classes. In this way, the performance of the learned classifier for the existing classes is improved.

2.1.3 Activity Recognition across Different Smart Homes

In the area of activity recognition across different smart homes, the sensor data in some existing smart home environment are used to facilitate the activity recognition in some new smart home environment. This is a process of transfer learning. Following [154], we refer to the existing smart home as the *source house*, the new smart home as the *target house*. As shown in Figure 1.3, it is a subarea of activity recognition in smart homes.

Existing works in this area are generally based on sensor networks consist of ambient sensors, and most of which are binary sensors. In some works, the sensor networks in the source and the target houses are different, while the activities aim to recognize are the same (activity recognition across smart homes with different sensor networks). In other works, both the sensor networks and the activities to recognize in the source and the target houses are different (activity recognition across smart homes with different sensor networks and label spaces). We review related works on these two aspects.

Activity recognition across smart homes with different sensor networks. In these works, the sensor networks in the source and the target houses are different. The instances in the feature space are extracted from the sensor readings. Following the general way in transfer learning, we refer to the feature space of the source-house instances as the *source feature space* and the feature space of the target-house instances as the *target feature space*. These two spaces are usually different. Methods addressing the problem of activity recognition across different feature spaces are proposed.

In some methods, during model learning, the testing instances are involved. For example, in [154], a set of “meta-features” are defined. With the training instances in the source feature space and the defined meta-features, prior of the parameters of the classifier (which is an HMM) in the target house is provided. Given the parameter prior, the HMM is learned in an iterative manner with the testing instances. In [153], a set of “function groups” (which is of the same function of meta-features) of the sensors is defined. With the function groups, mappings between the sensors in the source and the target houses are obtained. With the mapped sensors in the training and the testing datasets, an HMM is learned in an iterative manner.

In other methods, the testing instances are not involved in model learning. For example, in IFSR [50], meta-features got from labeled instances in both the source and the target feature spaces are used to learn the mapping from the target feature space to

the source feature space. A classifier is learned in the source feature space. In SHDA-RF [145], instances in the source feature space are mapped to the target feature space, and a classifier is learned in the target feature space. In [35], a set of “properties” (which is of the same function of meta-features) is used to get an alignment between the sensors in the source and the target houses. A classifier is learned with the aligned sensors. In UFSR [50], the meta-features are used to get a mapping from the target feature space to the source feature space, and a classifier is learned in the source feature space.

Activity recognition across smart homes with different sensor networks and label spaces. In these works, both the sensor networks and the label spaces in the source and the target houses are different. In HHTL [127], the mappings between sensors and activities in the source and the target houses are learned with the training and the testing instances in an iterative manner. Activity recognition is performed with the learned mappings. In [74], the testing instances in the target house are firstly classified to the classes in the source house, and then projected to the classes in the target house through the obtained web knowledge.

2.1.4 Multi-Resident Activity Recognition in Smart Homes

Activity recognition in smart homes is an area that has received much research [3, 112, 146, 155]. However, for the subarea of multi-resident activity recognition, as it is more challenging, there are fewer related works. In recent years, it obtains increasing attention. Benmansour et al. give a survey of existing works in [19].

In this area, some works are based on the wearable sensors. In [63, 160], wearable sensors are worn on the body of the experimental subjects, and activity recognition is based on the sensor readings. In these works, as each sensor is known beforehand which subject it is worn by, the sensor readings about different subjects can be separated naturally.

In most of existing works, as introduced in Section 2.1.1, the ambient sensors [13, 142] are adopted. As defined in [38], a sensor message together with the occurring time and the sensor ID can be defined as an “event”. Some works [20, 36, 73] perform activity recognition for each sensor event. In these works, when a sensor event is observed, they determine it belongs to which resident and which activity. Some of these works just focus on one aspect of the problem: in some works [40, 41], they just focus on determining an event belongs to which resident (this is referred to as data association); in other works [20, 36], the data association is assumed to be known, and they just focus on activity recognition. In other works, they do not just focus on one aspect and solve the whole problem. In [73], a CRF is used to solve the problem of data association and activity recognition jointly in an iterative manner. In [30], the Cartesian product of activity labels of different residents are used to form the combined labels. In this way, they convert the multi-resident activity recognition problem into a single-label classification problem, and learn an HMM or a CRF to classify the combined labels.

Besides the works performing activity recognition for each sensor event, there are works performing activity recognition based on fixed-length time segments. This is in accordance with many works in other sensor-based activity recognition problems [24, 155]. In these works, for a time segment, they determine the activities of each resident within this time segment. In [122], incremental decision trees are used for activity recognition. In [12], an FHMM and a tracking based approach are used.

In area of multi-resident activity recognition, there are also works utilizing active learning techniques [48], and works based on both the wearable sensors and the ambient sensors [140].

Almost all of the existing works are introduced in the situation of two residents (in the smart home, there are two residents), and are not evaluated in situations of more than two residents. However, most of the methods in these works are applicable to situations of more than two residents.

2.2 Zero-Shot Learning

Zero-shot learning is a learning paradigm in which the classes covered by the training instances and the classes we aim to classify are disjoint. The problems we studied in this thesis are under the problem setting of ARKN, in which zero-shot learning techniques are generally utilized. In this section, we review related works on zero-shot learning from three aspects: learning settings, semantic spaces, and methods. In the following of this section, we assume the feature space $\mathcal{X}$ is a D -dimensional real number space, i.e., $\mathbb{R}^D$; and the semantic space $\mathcal{T}$ is an M -dimensional real number space, i.e., $\mathbb{R}^M$.

2.2.1 Learning Settings

In zero-shot learning, the goal is to learn the classifier $f^u(\cdot)$ for the unseen classes. During model learning, if information about the testing instances is involved, the learned model is transductive for these specific testing instances. In zero-shot learning, this transduction can be embodied in two progressive degrees: transductive for specific unseen classes and transductive for specific testing instances. This is different from the well known transductive setting in semi-supervised learning [181], which is just for the testing instances. In the setting that is transductive for specific unseen classes, information about the unseen classes is involved in model learning, and the model is optimized for these specific unseen classes. In the setting that is transductive for specific testing instances, the transductive degree goes further. The testing instances are also involved in model learning, and the model is optimized for these specific testing instances. Based on the degree of transduction, we categorize zero-shot learning into three learning settings.

Definition 2.1 (Class-Inductive Instance-Inductive (CIII) Setting). Only labeled training instances D^{tr} and seen class prototypes T^s are used in model learning.

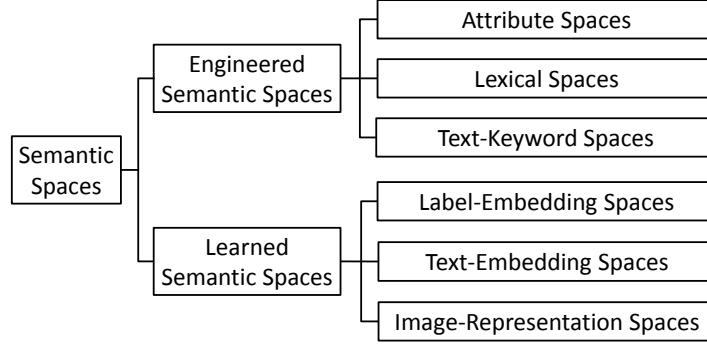


Figure 2.1: Different semantic spaces in zero-shot learning.

Definition 2.2 (Class-Transductive Instance-Inductive (CTII) Setting). Labeled training instances D^{tr} , seen class prototypes T^s , and unseen class prototypes T^u are used in model learning.

Definition 2.3 (Class-Transductive Instance-Transductive (CTIT) Setting). Labeled training instances D^{tr} , seen class prototypes T^s , unlabeled testing instances X^{te} , and unseen class prototypes T^u are used in model learning.

We can see, from CIII to CTIT, the classifier $f^u(\cdot)$ is learned with increasingly specific testing instances' information. In CIII, as no information about the testing instances is involved in model learning; while in CTII, the unseen class prototypes are involved in model learning; further, in CTIT, the models are optimized for specific unseen classes and testing instances. From CIII to CTIT, as increasing amount of information about the testing instances is involved in model learning, the performance of the learned model is generally getting better. However, the learned model is also increasingly restricted to certain unseen classes and testing instances.

2.2.2 Semantic Spaces

Semantic spaces contain semantic information about classes, and are an important part of zero-shot learning. Various semantic spaces have been exploited by existing works.

According to how a semantic space is constructed, we categorize them as: (1) engineered semantic spaces and (2) learned semantic spaces. A detailed taxonomy of the semantic spaces is shown in Figure 2.1.

2.2.2.1 Engineered Semantic Spaces

In engineered semantic spaces, each dimension of the semantic space is designed by humans. There are different kinds of engineered semantic spaces, each of which has a unique data source and a unique way to construct the space. Next, we introduce typical engineered semantic spaces that are used in zero-shot learning.

Attribute spaces. Attribute spaces are kinds of semantic spaces that are constructed by a set of *attributes*. They are one of the most widely used semantic spaces in zero-shot learning. In the pioneering works [90, 114] on zero-shot learning, attribute spaces have already been adopted. In an attribute space, a list of terms describing various properties of the classes are defined as attributes. Each attribute is usually a word or a phrase, corresponding to one property of these classes. For example, in the problem of daily living activity recognition, the attributes could be motion of the body (e.g., “translation motion” and “cyclic motion”), or time (e.g., “morning” and “noon”) [34]. Then, these attributes are used to form the semantic space, with each dimension being one attribute. For each class, the values of each dimension of the corresponding prototype are determined by whether this class has the corresponding attribute. Suppose in the problem of daily living activity recognition, there are three attributes: “posture upright”, “arm pendulum swing” and “translation motion”. For the class “running”, the corresponding values of these three attributes form the prototype $[1, 1, 1]$. For the class “standing and talking”, the corresponding values form the prototype $[1, 0, 0]$ [34]. In the above example, the attribute values are binary (i.e., 0/1). Thus, the resulting attribute space is referred to as a *binary attribute space*. In general, the attribute values can also be real numbers,

indicating the degree/confidence-level for one class having an attribute. The resulting attribute space is referred to as a *continuous attribute space*. Binary and continuous attribute spaces are the most common attribute spaces. In addition, there also exist *relative attribute spaces* [116], which measure the relative degree of having an attribute among different classes.

Lexical spaces. Lexical spaces are kinds of semantic spaces, which are constructed by a set of *lexical items*. Lexical spaces are based on the labels of the classes and datasets that can provide semantic information. The datasets can be some structured lexical databases, such as WordNet. Adopting WordNet as information source, there are different ways to construct semantic spaces. One way is by leveraging the hierarchical relationships in WordNet. Such as using all seen and unseen classes, together with their ancestors in WordNet to form the semantic space, with each dimension corresponds to one class [7, 8]. Then, for a class c_i , the value of the j th dimension of its prototype $\mathbf{t}_i$ is determined by the relation between class c_i and class c_j (the corresponding class of the j th dimension). In some approaches, the value is 1 if c_j is the ancestor of c_i or is c_i itself; otherwise, the value is 0 [7]. In other approaches, the value is determined by the distance between c_i and c_j in WordNet, the distance measure can be Jiang-Conrath distance [8], Lin distance [8], or path length similarity [8]. Besides hierarchical relation, other relationships such as partial relationships in WordNet can also be used to form the semantic space [138]. WordNet is the representation of general lexical databases. In some problems, there are problem-specific lexical databases. For example, in fine-grained named entity typing [99], there are predefined tree hierarchies of different entity types, which are used to form the semantic space. Along with the structured lexical databases, the datasets could also be some corpora. For example, in [114], each class is represented as a co-occurrence vector of the class label with the 5000 most frequent words from the Google Trillion-Word-Corpus.

Text-keyword spaces. Text-keyword spaces are kinds of semantic spaces, which are constructed by a set of keywords extracted from the text descriptions of each class. In text-keyword spaces, the most common source of the text descriptions is the websites, including both general websites like Wikipedia [8, 123], and domain specific websites. For example, in [17, 46], as the task is zero-shot flower recognition in images, both Plant Database and Plant Encyclopedia are used (which are specific for plants) to obtain the text descriptions for each flower class. In addition to predefined websites, such text descriptions can also be obtained from search engines. For example, in [180], each class name is used as a query to Google to find the web pages describing the class. In some specific problems, there are specific approaches to obtain the text descriptions. Such as in zero-shot video event detection [42], the text descriptions of the events can be obtained from the event kits provided in the dataset. After obtaining the text descriptions for each class, the next step is to construct the semantic space and generate class prototypes from these descriptions. The semantic space is usually constructed by the keywords extracted from these text descriptions, with each dimension corresponds to a keyword. To construct prototypes of the classes, different approaches are proposed. Some works use the binary occurrence indicator [123] or Bag of Words (BOW) representation [8] for each text description. On the other hand, some works exploit information retrieval techniques. For example, [17, 47, 180] use term frequency inversed document frequency (TF-IDF) to represent each class, whereas [46] applies Clustered Latent Semantic Indexing algorithm on the TF-IDF vectors to obtain representation vectors with reduced dimensions.

2.2.2.2 Learned Semantic Spaces

In learned semantic spaces, the dimensions in the spaces are not designed by humans. Prototypes of each class are obtained from the output of some machine learning models. In these prototypes, each dimension does not have an explicit semantic meaning. Instead,

the semantic information is contained in the whole prototype. The models used to extract the prototypes can be pretrained in other problems, or trained specifically for the zero-shot learning problem. Next, we introduce typical learned semantic spaces used in zero-shot learning.

Label-embedding spaces. Label-embedding spaces are kinds of semantic spaces in which the class prototypes are obtained through the embedding of class labels. This kind of space is introduced in view of the development and wide utilization of word embedding techniques in natural language processing. In word embedding, words or phrases are embedded into a real number space as vectors. This embedding space contains semantic information. In this space, semantically similar words or phrases are embedded as nearby vectors, while semantically dissimilar words or phrases are embedded as vectors far apart. In zero-shot learning, for each class, the class label of it is a word or a phrase. Thus, it can be embedded into a word embedding space, with the obtained corresponding vector as the prototype. In existing works, different embedding techniques, such as Word2Vec [8, 159, 165] and GloVe [8, 165], have been adopted. Besides, different corpora, from general ones such as Wikipedia [143, 159], to specific ones such as texts from Flickr [25], have been used to learn the embedding model. In addition to generating one prototype for each class, there are also works [105, 132] that generate more than one prototype for each class in the label embedding space. In these works, the prototypes of a class are usually multiple vectors following Gaussian distribution.

Text-embedding spaces. Text-embedding spaces are kinds of semantic spaces in which the class prototypes are obtained by embedding the text descriptions for each class. Being similar to text-keyword spaces, the semantic information in text-embedding spaces also comes from text descriptions. However, there exists a major difference between these two kinds of spaces. Specifically, a text-keyword space is constructed through extracting keywords and using each of them as a dimension in the constructed space. A text-embedding space is constructed through some learning models. The text descriptions

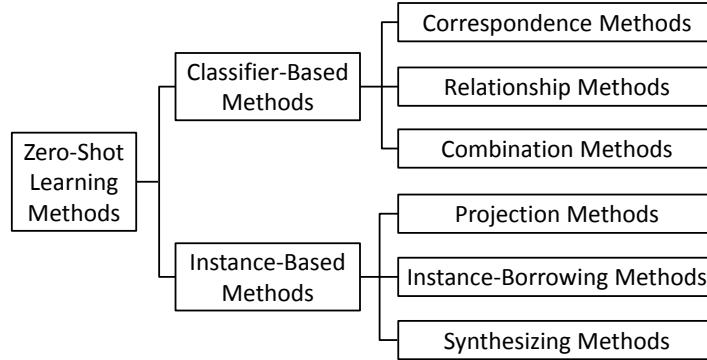


Figure 2.2: Different categories of methods in zero-shot learning.

of each class are used as the input of the learning model, and the output vectors are regarded as the prototypes of this class. For example, in an image object recognition task [129], several text descriptions are collected for each class. These text descriptions are used as input of a text encoder model, and the output vectors are regarded as the class prototypes.

Image-representation spaces. Image-representation spaces are kinds of semantic spaces, in which the class prototypes are obtained from images belonging to each class. For example, in a video action recognition task [161], images of different action classes are obtained through the search engines. Then, for each action class, images belonging to this class are used as input to some pretrained model (e.g., GoogLeNet pretrained on the ImageNet dataset). The output vectors from the model are combined to form a representation vector and it is used as the prototype of this action class.

2.2.3 Methods

We classify existing zero-shot learning methods into two categories: *classifier-based methods* and *instance-based methods*. For classifier-based methods, the focus is on how to directly learn a classifier for the unseen classes. For instance-based methods, the focus is on how to obtain labeled instances belonging to the unseen classes and use them for

classifier learning. We summarize a method categorization hierarchy in Figure 2.2. Next, we introduce each category in this method hierarchy. When introducing the methods, we mainly focus on the most standard data setting of zero-shot learning. That is, for a zero-shot learning task, we consider one semantic space and represent each class with one prototype in that space.

2.2.3.1 Classifier-Based Methods

According to the approaches to construct classifiers, we further classify the classifier-based methods into three subcategories: (1) *correspondence methods*, (2) *relationship methods*, and (3) *combination methods*. Existing classifier-based methods usually take a one-vs-rest solution for learning the multi-class zero-shot classifier $f^u(\cdot)$. That is, for each unseen class c_i^u , they learn a binary one-vs-rest classifier. We denote $f_i^u(\cdot) : \mathbb{R}^D \rightarrow \{0, 1\}$ as the binary one-vs-rest classifier for class $c_i^u \in \mathcal{U}$. Therefore, the eventual zero-shot classifier $f^u(\cdot)$ for the unseen classes consists of N_u binary one-vs-rest classifiers $\{f_i^u(\cdot) | i = 1, \dots, N_u\}$.

Correspondence methods. This kind of methods aims to construct the classifier for unseen classes via the correspondence between the binary one-vs-rest classifier for each class and its corresponding class prototype. In the semantic space, for each class, there is just one corresponding prototype. Thus, this prototype can be regarded as the “representation” of this class. Meanwhile, in the feature space, for each class, there is a corresponding binary one-vs-rest classifier, which can also be regarded as the “representation” of this class. Correspondence methods aim to learn a *correspondence function* between these two types of “representations”. In correspondence methods, the correspondence function $\varphi(\cdot; \varsigma)$ takes the prototype $\mathbf{t}_i$ of a class c_i as input, and outputs the parameter ω_i of the binary one-vs-rest classifier $f_i(\cdot; \omega_i)$ for this class, i.e., $\omega_i = \varphi(\mathbf{t}_i; \varsigma)$. After obtaining this correspondence function, for an unseen class c_i^u , with its prototype $\mathbf{t}_i^u$, the corresponding binary one-vs-rest classifier $f_i^u(\cdot)$ can be constructed.

The general procedure of correspondence methods is as follows. Firstly, with the available data (under different learning settings, the available data are different), the correspondence function $\varphi(\cdot; \varsigma)$ is learned. Then, for each unseen class c_i^u , with its prototype $\mathbf{t}_i^u$ and the learned correspondence function $\varphi(\cdot; \varsigma)$, the binary one-vs-rest classifier $f_i^u(\cdot)$ is constructed. Finally, with these binary classifiers $\{f_i^u(\cdot)\}_{i=1}^{N_u}$ for unseen classes, classification of the testing instances X^{te} is achieved.

Relationship methods. This kind of methods aims to construct classifier for the unseen classes based on the relationships among classes. In the feature space, binary one-vs-rest classifiers $\{f_i^s(\cdot)\}_{i=1}^{N_s}$ for the seen classes can be learned with the available data (under different learning settings, the available data are different). Meanwhile, the relationships among the seen and the unseen classes can be obtained by calculating the relationships among corresponding prototypes or by other approaches. Relationship methods aim to construct the classifier $f^u(\cdot)$ for the unseen classes through these learned binary seen class classifiers and these class-relationships.

The general procedure of relationship methods is as follows. Firstly, binary one-vs-rest classifiers $\{f_i^s(\cdot)\}_{i=1}^{N_s}$ for the seen classes $\mathcal{S}$ are learned with the available data. Then, the relationships δ among the seen and the unseen classes are calculated via the corresponding class prototypes or obtained through other approaches. Finally, with these classifiers $\{f_i^s(\cdot)\}_{i=1}^{N_s}$ and the relationships δ , classifier $f^u(\cdot) = \{f_i^u(\cdot)\}_{i=1}^{N_u}$ for the unseen classes $\mathcal{U}$ is constructed and classification of the testing instances X^{te} is achieved.

Combination methods. This kind of methods aims to construct the classifier for unseen classes by the combination of classifiers for basic elements which are used to constitute the classes. In combination methods, it is regarded that there is a list of “basic elements” to constitute the classes. Each of the seen and the unseen classes is a combination of these basic elements. Embodied in the semantic space, it is regarded that each dimension represents a basic element, and each class prototype denoting the combination of these

basic elements for the corresponding class. Thus, methods in this category are mainly suitable for semantic spaces in which each dimension of the class prototypes takes the value of either 1 or 0, denoting whether a class has the corresponding element or does not. Typical examples of these kinds of spaces are binary attribute spaces, some lexical spaces, and some text-keyword spaces.

The general procedure of combination methods is as follows. With the available data (the available data are different under different learning settings), classifiers $\{f_i^a(\cdot)\}_{i=1}^M$ for the attributes are firstly learned. Then, with the learned classifiers for the attributes, classifier $f^u(\cdot) = \{f_i^u(\cdot)\}_{i=1}^{N_u}$ for the unseen classes is obtained through some inference model.

2.2.3.2 Instance-Based Methods

Instance-based methods aim to firstly obtain labeled instances for the unseen classes, then with these instances to learn the zero-shot classifier $f^u(\cdot)$. According to the source of these instances, existing instance-based methods can be classified into three subcategories: (1) *projection methods*, (2) *instance-borrowing methods*, and (3) *synthesizing methods*.

Projection methods. This kind of methods aims to obtain labeled instances for the unseen classes by projecting both the feature space instances and the semantic space prototypes into a common space. In the feature space, there are labeled training instances belonging to the seen classes. Meanwhile, in the semantic space, there are prototypes of both the seen and the unseen classes. Both of the feature space and the semantic space are real number spaces, with instances and prototypes as vectors in them. In this view, the prototypes can also be regarded as labeled instances. Thus, we have labeled instances in two spaces (the feature space $\mathcal{X}$ and the semantic space $\mathcal{T}$). In projection methods, instances in these two spaces are projected into a common space. In this way, we can obtain labeled instances belonging to the unseen classes. In this paper, we refer to this common space as the *projection space*, denoted as $\mathcal{P}$.

The general procedure of projection methods is as follows. Firstly, instances $\mathbf{x}_i$ in the feature space $\mathcal{X}$ and prototypes $\mathbf{t}_j$ in the semantic space $\mathcal{T}$ are projected into the projection space $\mathcal{P}$ with the projection functions $\theta(\cdot)$ and $\xi(\cdot)$ respectively: $\mathcal{X} \rightarrow \mathcal{P} : \mathbf{z}_i = \theta(\mathbf{x}_i)$ and $\mathcal{T} \rightarrow \mathcal{P} : \mathbf{b}_j = \xi(\mathbf{t}_j)$. Then, classification is performed in the projection space. In projection methods, for each unseen class c_i^u , it does not have labeled instances in the feature space, thus its prototype $\mathbf{t}_i^u$ in the semantic space is the only labeled instance belonging to this class. That is, for each unseen class, there is only one labeled instance available. It is difficult to learn classifiers like SVM or logistic regression classifier with so few labeled instances for the unseen classes. As a result, in existing projection methods, the classification is usually performed by *nearest neighbor classification (1NN)* or some variants of it. The 1NN method can work in situations where just one labeled instance is available for the classes needed to classify, which is just suitable for this scenario. The 1NN method is a kind of lazy learning method, with no explicit learning process needed. Thus, when use it for classification, no explicit process of learning classifiers for the unseen classes is needed.

Instance-borrowing methods. This kind of methods aims to obtain labeled instances for the unseen classes by borrowing from the training instances. Instance-borrowing methods are based on the similarities between classes. For example, when recognizing the activities, suppose we want to learn a classifier for the class “jogging”, but do not have the corresponding labeled instances. However, we have some labeled instances belonging to classes “running” and “walking”. As they are similar activities to “jogging”, when learning classifier for the class “jogging”, we can use instances belonging to these two classes as positive instances. This kind of methods follows the way human beings recognize the world [67]. We may have never seen instances belonging to some classes, but have seen instances belonging to some similar classes. With the knowledge of these similar classes, we are able to recognize instances belonging to the unseen classes.

Table 2.1: Some representative zero-shot learning methods belonging to different categories and under different learning settings.

Method Category		CIII	CTII	CTIT
Classifier-based methods	Correspondence methods	[5, 6, 8, 52, 129, 139, 165]	[46, 119, 163]	[68, 94, 95, 157]
	Relationship methods	[56, 77, 87, 137]	[43, 55]	[168, 179]
	Combination methods	[57, 90, 93, 96, 151]	—	[59, 136]
Instance-based methods	Projection methods	[79, 86, 114, 124, 143, 176]	[78, 162]	[53, 85, 162, 169, 170, 175]
	Instance-borrowing methods	—	[58, 66, 67, 180]	[66, 67]
	Synthesizing methods	—	[23, 65, 156, 159, 167]	—

The general procedure of instance-borrowing methods is as follows. Firstly, for each unseen class c_i^u , some instances from the training instances are borrowed and assigned the label of this class. Then, with the borrowed instances for all unseen classes, the classifier $f^u(\cdot)$ for the unseen classes is learned, and classification of the testing instances X^{te} is achieved.

Synthesizing methods. This kind of methods aims to obtain labeled instances for the unseen classes by synthesizing some pseudo instances.

The general procedure of synthesizing methods is as follows. Firstly, for each unseen class c_i^u , some pseudo labeled instances are synthesized. Then, with these synthesized instances for all unseen classes, classifier $f^u(\cdot)$ for the unseen classes is learned and classification of the testing instances X^{te} is achieved.

2.2.3.3 Others

We summarize some representative zero-shot learning methods belonging to different methods and under different learning settings in Table 2.1. In the following, we introduce

two different problem settings in zero-shot learning: multi-label zero-shot learning and generalized zero-shot learning.

Multi-label zero-shot learning. In the above introduction, we focus on the setting in which each instance belongs to one class, i.e., has one class label. However, there are some problems in which each instance can have more than one class label. This problem setting is usually referred to as *multi-label zero-shot learning*. When learning the classification model, existing methods either treat each class label individually [25, 99, 102, 173], or exploit multi-label correlations [54, 92, 106, 133, 178].

Generalized zero-shot learning. In zero-shot learning, classes covered by the training and the testing instances are disjoint. In practice, this problem setting is somewhat restricted. In many applications, not only instances belonging to the unseen classes, but also instances belonging to the seen classes can appear in the testing phase. Under the setting of *generalized zero-shot learning* [28], the testing instances can come from both the seen and the unseen classes. Due to the co-existence of the seen and the unseen classes during the testing phase, problems under this setting are more challenging. A number of zero-shot learning methods [166] have also been tested under the generalized zero-shot learning setting. However, their performance currently is not as good as that under the setting of zero-shot learning [166]. More methods that can achieve better performance under the setting of generalized zero-shot learning are needed to support wider practical applications.

2.2.4 Zero-Shot Learning in ARKN Problems

Under the setting of ARKN, no labeled instance for the unseen classes is available. Zero-shot learning methods are generally utilized to address the problems.

In [34], the problem is activity recognition based on wearable sensor readings. The semantic space used is the binary attribute space. The proposed method belongs to

the projection methods under CIII setting. Through the classifier for each attribute, the instances in the feature space are projected into the semantic space. Then 1NN is performed in the semantic space for classification. In [33], the sequential information is considered, and CFRs are used for attribute prediction model. In [111], the importance value for each attribute is considered when performing the 1NN classification. In [9], the problem setting is activity recognition in the factory environment. The defined “objects” and “actions” act as the role of attributes. Classifiers for them are learned. Then, classification is based on the defined relation between the activities, the objects and the actions. In [100], the label-embedding vectors and the Twitter timestamps are used as the prototypes. In CDAR [180], the semantic information is got from the web texts. With this information, pseudo instances belonging to the unseen classes are obtained by borrowing from the training instances and used to learn a weighted SVM. This method is under the CTII setting.

Chapter 3

Unseen Activity Recognition for One Person

In sensor-based activity recognition, the previously unseen activities can often be encountered. In this chapter, we study the problem in which the activities covered by the training data and the activities needed to recognize are disjoint, i.e., the activities needed to recognize belong to the unseen classes. It is a zero-shot learning problem. We propose a method to address this problem. The proposed method learns a compatibility function between the instances in the feature space and the prototypes in the semantic space. With this function, a testing instance is classified to the unseen class with the highest compatibility score. Through extensive experiments, we demonstrate the effectiveness of our method.

This chapter is organized as follows. In Section 3.1, we give an introduction about the background of the problem. In Section 3.2, we describe the proposed method. In Section 3.3, we present the experiments. Finally, in Section 3.4, we give a summary.

3.1 Background

Sensor-based activity recognition is an area that has received extensive research in the past years [24, 29, 83]. In this area, many recognition methods have been proposed, and

inspiring performance has been achieved. Despite of this, there still exist some limitations in these methods. One prominent limitation is the ability to recognize the previously unseen activities. In this area, most of existing methods can only recognize activities which have labeled instances in the training data. For example, if the training data cover N activity classes, the learned classifier is generally an N -way classifier for these activities, and it does not have the ability to recognize the previously unseen activities. However, in practical applications, we can often encounter activities which are not covered by the training data, and we also need to recognize them. There are several reasons leading this phenomenon. From the aspect of activities, on the one hand, the number of activities humans can perform is large [10, 34]; on the other hand, with the change of the time and the occasions, the activities needed to recognize can also change [2]. From the aspect of data collecting, the process of collecting labeled sensor readings is time consuming and costly [10, 34, 172]. When learning an activity recognition model, in the training phase, it is hard to collect labeled sensor readings for all of the activities we can encounter when using it for activity recognition. Based on this, how to recognize the previously unseen activities is a problem of practical value, but cannot be solved by most of existing methods [24, 89, 91].

3.1.1 Problem Description

Under the problem setting of sensor-based activity recognition including unseen classes, we focus on the problem in which, for one person, the activity classes covered by the training and the testing data are disjoint. In the training phase, we do not have the testing sensor readings which are needed to classify, and we do not know the labels of the unseen classes. The set of the unseen classes are known just in the testing phase.

This problem can occur in many applications. In some applications, the environments of activity recognition in the training and the testing phases can be different. For example, we may have some labeled sensor readings collected for a person in the working

environment. With these data, we can learn a classifier. However, when we use the classifier for activity recognition, the environment could change. We may need to perform activity recognition for this person in the home environment. In different environments, the activities needed to recognize are generally different. Thus, the activities covered by the training and the testing data can be disjoint. Also during model learning, we do not know the labels of the unseen classes, and we do not have the testing sensor readings that are needed to classify. The set of the activity classes in the home environment is determined just during the testing phase. In some applications, even the environment of activity recognition is not changed, the set of activities covered by the training and the testing data can also be different. For example, in the home environment, in the data collecting stage, we may be interested in the activities related with house cleaning, and the collected labeled data are about the relevant activities; however, when we perform activity recognition, the activities of interest can change, and we may need to recognize activities related with cooking and having meals. Being similar to the above example, in this case, the activities covered by the training and the testing data can also be disjoint. In addition, during model learning, no information about the testing data is available, and the set of unseen classes is determined in the testing phase.

Based on the descriptions given above, we can give a statement of the problem as follows. In the training phase, we have labeled sensor readings collected by a set of sensors. The sensor readings belong to the seen classes, $\mathcal{S} = \{c_i^s | i = 1, \dots, N_s\}$, where N_s is the number of seen classes. In the testing phase, we have the unlabeled sensor readings got from the same set of sensors, and the sensor readings belong to the unseen classes, $\mathcal{U} = \{c_i^u | i = 1, \dots, N_u\}$, where N_u is the number of unseen classes. The goal is that, with the labeled sensor readings belonging to the seen classes, we learn a classification model which can classify the unlabeled sensor readings belonging to the unseen classes.

According to Definition 1.3, this problem is a zero-shot learning problem. Specifically, according to Definition 2.1, this problem is a zero-shot learning problem under the CIII

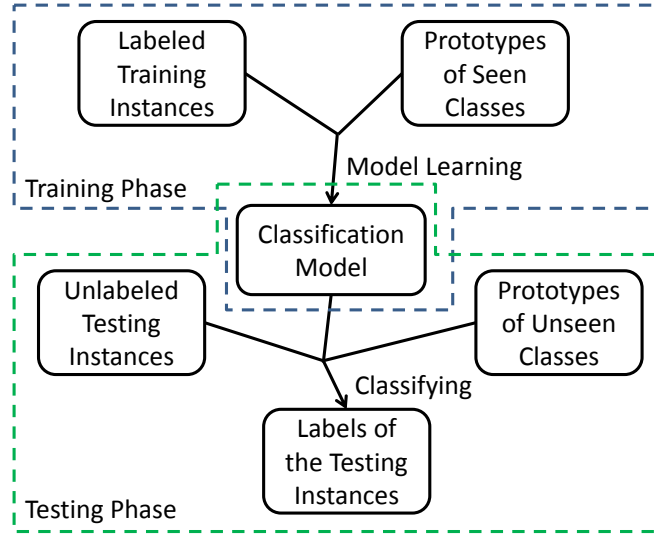


Figure 3.1: General process of methods for zero-shot learning problems under CIII learning setting.

learning setting. In this problem, the seen classes are covered by the training data, and the unseen classes are covered by the testing data. The seen and the unseen classes are disjoint. In the training phase, just information about the seen classes and the training instances is utilized. The set of unseen classes are known in the testing phase.

3.1.2 General Process and Existing Works

General process. As stated above, this problem is a zero-shot learning problem under the CIII learning setting. The general process of existing methods on zero-shot CIII-setting problems is shown in Figure 3.1. In the training phase, besides the labeled training instances, the prototypes of the seen classes (contain semantic information about the seen classes) are also involved. The classification model is learned with these data. In the testing phase, the learned classification model along with the prototypes of the unseen classes (contain semantic information about the unseen classes) are utilized for the classification of the testing instances.

The involvement of the semantic information makes the recognition of the unseen

activity classes being possible. As this problem is under the CIII learning setting, in this process, just the semantic information about the seen classes is involved in the training phase, and the semantic information about the unseen classes is involved just in the testing phase. The involved semantic information should be effectively utilized in the processes of model-learning and testing-instance-classification.

Existing works. Most of existing works on zero-shot CIII-setting problems are proposed in the area of computer vision and proposed for problems like image recognition, image retrieval, or video recognition. There are limited amount of methods [33, 34] proposed in the area of sensor-based activity recognition. In [34], the binary attribute space is adopted as the semantic space. In the training phase, with the training instances and the seen class prototypes, the classifiers for each attribute are learned. In the testing phase, the testing instances are projected into the semantic space, and 1NN classification is performed. However, in this method, the model is optimized for each dimension of the semantic space (each attribute) independently, not for the whole classification process. In [33], sequential information is added in model learning, however, the model optimization is still for each attribute independently.

3.2 Proposed Method

In this chapter, we propose a method for zero-shot CIII-setting problems in the area of sensor-based activity recognition. Our method learns a compatibility function between the feature space and the semantic space, and this compatibility function is used for testing instance classification.

3.2.1 Data Preprocessing

For the raw sensor readings, we need to extract instances from them. In addition, in zero-shot learning, the semantic information is necessary. Firstly, we perform feature extraction and semantic information involvement.

Feature extraction. The sensor readings are time series data. When dealing with these data, we adopt the widely used sliding window strategy [24] to separate them into a collection of segments. Each segment is considered as an independent instance. Features of each instance are extracted from the sensor readings in the corresponding segment.

Semantic information involvement. We adopt the binary attribute space as the semantic space. As introduced in Section 2.2.2.1, in binary attribute space, each prototype is a 0-1 vector. If the instances of a class have an attribute, the corresponding dimension of the prototype is of value “1”; otherwise, it is of value “0”. In our method, we adopt the binary attribute space as the semantic space. This is because, in our problem, the problem setting is sensor-based activity recognition. The attributes and the prototypes can be defined by the common sense. Thus, it is not difficult to obtain them. In addition, it has been validated in [33, 34] that the binary attribute space is effective in sensor-based zero-shot activity recognition problems.

3.2.2 Problem Formulation

After the processes of feature extraction and semantic information involvement, the problem can be formalized as follows.

- In the training phase, we have N_{tr} labeled training instances $\{(\mathbf{x}_i^{tr}, y_i^{tr})\}_{i=1}^{N_{tr}}$. For each instance $(\mathbf{x}_i^{tr}, y_i^{tr})$, $\mathbf{x}_i^{tr}$ is the instance in the feature space $\mathcal{X}$, and y_i^{tr} is the class label, which belongs to the seen classes $\mathcal{S}$. In the semantic space $\mathcal{T}$, for each seen class c_i^s , there is a corresponding prototype $\mathbf{t}_i^s$.
- In the testing phase, we have N_{ts} testing instances $\{\mathbf{x}_i^{ts}\}_{i=1}^{N_{ts}}$ in the feature space $\mathcal{X}$. These instances belong to the unseen classes $\mathcal{U}$. In the semantic space $\mathcal{T}$, for each unseen class c_i^u , we have the corresponding prototype $\mathbf{t}_i^u$.

The activity classes covered by the training and the testing instances are disjoint, i.e., $\mathcal{S} \cap \mathcal{U} = \emptyset$. We assume the feature space is a D -dimensional real number space, i.e.,

$\mathbb{R}^D$, the semantic space is an M -dimensional real number space, i.e., $\mathbb{R}^M$. The goal is to learn a classifier $f(\cdot)$ that can predict class labels of the testing instances.

3.2.3 Compatibility Based Methods

Our proposed method belongs to the category of compatibility based zero-shot learning methods. In this section, we give an introduction of this kind of method.

The motivation of compatibility based zero-shot learning methods [5, 6, 8, 129, 165] is to learn a *compatibility function* between the feature space and the semantic space. Compatibility function is a kind of function widely used in structured prediction problems [148]. In structured prediction problems, the output that are needed to predict are structured objects, such as the sequences and the strings. The compatibility function takes an instance and a structured-object as input, and outputs a score denoting the degree of compatibility. For a testing instance, the predicted structured-object is the object with the highest compatibility score.

In recent years, the compatibility function is also widely used in zero-shot learning problems [5, 6, 8, 129, 165]. In existing compatibility based zero-shot learning methods under the CIII learning setting, the process is generally as follows:

- In the training phase, with the labeled training instances $\{(\mathbf{x}_i^{tr}, y_i^{tr})\}_{i=1}^{N_{tr}}$ and the seen class prototypes $\{\mathbf{t}_j^s\}_{j=1}^{N_s}$, we can learn a compatibility function $F(\mathbf{x}_i, \mathbf{t}_j) : \mathcal{X} \times \mathcal{T} \rightarrow \mathbb{R}$. This function takes a feature space instance $\mathbf{x}_i$ and a semantic space prototype $\mathbf{t}_j$ as input, and outputs a real number (compatibility score) indicating the degree of compatibility. If the instance and the prototype come from the same class, the value of the compatibility score is high; otherwise, the value of the score is low.
- In the testing phase, for a testing instance $\mathbf{x}_i^{te}$, we calculate its compatibility score with all unseen class prototypes $\{\mathbf{t}_j^u\}_{j=1}^{N_u}$, and classify it to the unseen class with the highest compatibility score.

In zero-shot learning problems, the prototypes are the structured-objects. The compatibility based zero-shot learning methods utilize the compatibility function to perform the testing instance classification.

Bilinear function. In existing widely used methods [6, 8, 52, 139], most of them utilize the *bilinear function* as the compatibility function, i.e., in these methods, the compatibility function is in form of

$$F(\mathbf{x}_i, \mathbf{t}_j) = \mathbf{x}_i^T W \mathbf{t}_j, \quad (3.1)$$

where $W \in \mathbb{R}^{D \times M}$ is the parameter needed to learn.

One-vs-rest classifiers. The compatibility based methods described above can be seen from another view [139]. In these methods, given the compatibility function $F(\cdot, \cdot)$, for a class c_j and its corresponding prototype $\mathbf{t}_j$, the one-vs-rest classifier for this class c_j can be constructed as $f_j(\cdot) = F(\cdot, \mathbf{t}_j)$. From this view, in the training phase, with the prototypes of the seen classes, the one-vs-rest classifiers for the seen classes are constructed. In the testing phase, with the learned compatibility function and the prototypes of the unseen classes, one-vs-rest classifiers for the unseen classes are constructed. The compatibility function can be seen as the common part of these one-vs-rest classifiers.

Thus, the compatibility based methods belong to the correspondence methods introduced in Section 2.2.3, which aims to construct the classifier for the unseen classes through the correspondence between the binary one-vs-rest classifier for each class and its corresponding prototype. The learned compatibility function $F(\cdot, \cdot)$ acts as the correspondence function in the correspondence methods.

As stated above, most of the widely used compatibility based methods utilize the bilinear function as the compatibility function. With the bilinear compatibility function, the one-vs-rest classifier constructed for a class c_j is in form of $F(\cdot, \mathbf{t}_j) = W \mathbf{t}_j$, which is a *linear classifier*. Thus, by bilinear compatibility function, the one-vs-rest classifiers

obtained are linear classifiers. As an extension, in [165], piecewise bilinear function is used as the compatibility function. By piecewise bilinear compatibility function [165], the obtained one-vs-rest classifiers are piecewise-linear classifiers. However, in view of the classifiers, the classification ability of these linear and piecewise-linear classifiers is limited. In many cases, instances in the feature space are not linearly separable, and also hard to be separated by piecewise-linear classifiers. To separate these instances, some more complex nonlinear classifiers are needed.

In category of compatibility based methods, there have been some methods proposed to learn nonlinear classifiers. However, the classification model learned in these methods are specific for certain data types (images [5, 17] or texts [5, 129]), and are not suitable for our problem.

3.2.4 Proposed NCBM Method

Existing compatibility based methods are generally proposed in the area of computer vision, for the problems of image recognition [5, 6, 8] or image retrieval [129]. The effect of this kind of method has not been verified in sensor-based zero-shot activity recognition problems. In this chapter, we propose a compatibility based method, which aims to learn nonlinear one-vs-rest classifiers for different classes. As our method belongs to compatibility based methods, and nonlinear one-vs-rest classifiers are obtained, we refer to it as *Nonlinear Compatibility Based Method (NCBM)*. As shown in Figure 3.2, the process of our method can be divided into two steps.

Step 1: Nonlinear projection. In this step, for an instance $\mathbf{x}_i$ in the feature space, we project it into a hidden space $\mathcal{H}$ (which is a real number space of dimension H , i.e., $\mathbb{R}^H$), and get a new representation $\mathbf{h}_i$ of it.

$$\mathbf{h}_i = g(U\mathbf{x}_i + \mathbf{b}), \quad (3.2)$$

where $g(\cdot)$ is the nonlinear function, $U \in \mathbb{R}^{H \times D}$ and $\mathbf{b} \in \mathbb{R}^H$ are the parameters.

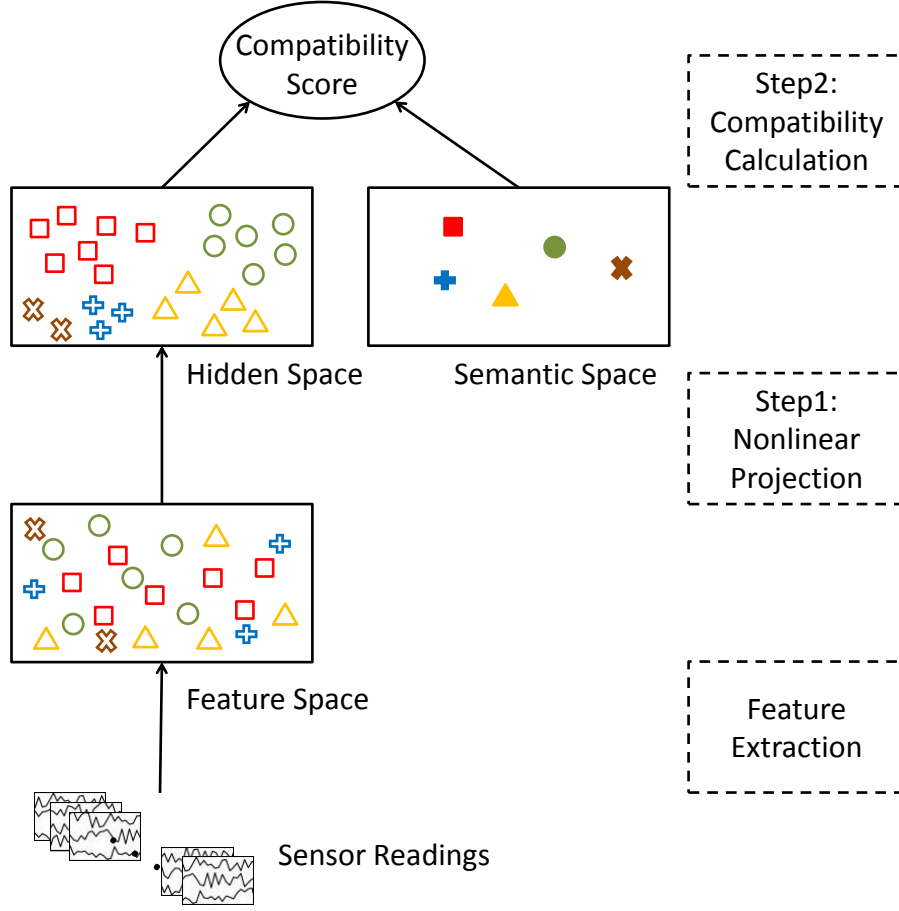


Figure 3.2: Process of the NCBM method.

Step 2: Compatibility calculation. In this step, we learn a compatibility function $F(\cdot, \cdot)$ between the instances in this hidden space $\mathcal{H}$ and the prototypes in the semantic space $\mathcal{T}$.

$$F(\mathbf{h}_i, \mathbf{t}_j) = \mathbf{h}_i^T W \mathbf{t}_j, \quad (3.3)$$

where $W \in \mathbb{R}^{H \times M}$ is the parameter.

In view of the one-vs-rest classifiers, for a class c_j , the one-vs-rest classifier we learned in this way is in form of $F(\cdot, \mathbf{t}_j) = g(\cdot)^T W \mathbf{t}_j$. They are nonlinear classifiers for the instances, and are more capable than linear classifiers.

Objective function. To learn this classification model, with the training instances

$\{(\mathbf{x}_i^{tr}, y_i^{tr})\}_{i=1}^{N_{tr}}$ and the seen class prototypes $\{\mathbf{t}_i^s\}_{i=1}^{N_s}$, we optimize the objective function in form of

$$\min \frac{1}{N_{tr}} \frac{1}{N_s} \sum_{i=1}^{N_{tr}} \sum_{j=1}^{N_s} \max\{0, 1 - \mathbb{I}_{ij} F(g(\mathbf{x}_i^{tr}), \mathbf{t}_j^s)\}, \quad (3.4)$$

where $\mathbb{I}_{ij}$ is an indicator that evaluates to 1 if $\mathbf{x}_i^{tr}$ and $\mathbf{t}_j^s$ come from the same class, and -1 otherwise. We use hinge loss as the loss function when learning the model.

Optimization. To optimize the objective function, we use a stochastic gradient descent based method Adagrad [45] to optimize the parameters. The optimization is performed in a mini-batch manner. When learning the model, from the objective function, we use the backpropagation strategy to update the parameters. The training process is performed to a fixed number of epochs. The training phase (model learning phase) of the NCBM method is summarized in Algorithm 1.

Algorithm 1 Training phase of the NCBM method

Input: Labeled training instances $\{(\mathbf{x}_i^{tr}, y_i^{tr})\}_{i=1}^{N_{tr}}$, seen class prototypes $\{\mathbf{t}_i^s\}_{i=1}^{N_s}$, number of training epochs P , mini-batch size bz , initial learning rate η .

Output: W , U , and $\mathbf{b}$.

- 1: Initialize the parameters W , U , and $\mathbf{b}$, initialize the learning rate of the Adagrad method to be η ;
 - 2: **for** $p = 1$ to P **do**
 - 3: **for** $k = 1$ to $K = \lfloor N_{tr}/bz \rfloor$ **do**
 - 4: With the training instances $\{(\mathbf{x}_i^{tr}, y_i^{tr})\}_{i=(k-1) \times bz + 1}^{k \times bz}$ in the k th batch, and the seen class prototypes $\{\mathbf{t}_i^s\}_{i=1}^{N_s}$, through Equations 3.2 and 3.3, calculate the objective in Equation 3.4 /* as mini-batch is used for optimization, thus N_{tr} in Equation 3.4 is replaced to bz accordingly */;
 - 5: Update the parameters W , U , and $\mathbf{b}$ through backpropagation /* the parameters for the Adagrad method are also updated */;
 - 6: **end for**
 - 7: **end for**
 - 8: **return** W , U , and $\mathbf{b}$.
-

Testing instance classification. After the parameters W , U , and $\mathbf{b}$ in the classification model being learned, for an unseen class c_j^u , the corresponding one-vs-rest classifier can be constructed as $f_j(\cdot) = F(\cdot, \mathbf{t}_j^u)$. The classifier $f(\cdot)$ for the unseen classes $\mathcal{U}$ consists

of N_u binary one-vs-rest classifiers $\{f_j(\cdot) | j = 1, \dots, N_u\}$ for each of the unseen class. For a testing instance $\mathbf{x}_i^{te}$, it is classified to the unseen class with the highest compatibility score,

$$f(\mathbf{x}_i^{te}) = \arg \max_{c_j^u \in \mathcal{U}} F(\mathbf{x}_i^{te}, \mathbf{t}_j^u). \quad (3.5)$$

This is equivalent to classifying the instance $\mathbf{x}_i^{te}$ to the unseen class the one-vs-rest classifier of which has the highest prediction score.

3.3 Experiments

3.3.1 Datasets

To evaluate our method, we use three publicly available datasets in the experiments. These three datasets are widely used in sensor-based activity recognition problems. Each dataset can be seen as a representation of one kind of dataset got by a certain data collecting manner. PAMAP2 is the dataset got from the experimental setting. OPP represents the dataset got from a certain scenario. TUD is the representation of datasets got from the daily living. Datasets PAMAP2 and OPP are got from UCI Machine Learning Repository [44]. In the following, we give an introduction of these datasets.

PAMAP2 [131]: PAMAP2 Physical Activity Monitoring Dataset (PAMAP2)¹ contains sensor readings of 18 activities performed by 9 participants. Each individual is asked to perform the activities, and the corresponding sensor readings are recorded. In this dataset, one heart rate monitor and three inertial measurement units (IMUs) are used for data collecting. The IMUs are positioned over the wrist on the dominant arm, on the chest, and on the dominant side's ankle. Each IMU contains two accelerometers, one gyroscope and one magnetometer. Sampling frequency of the heart rate monitor is about 9Hz, and that of all other sensors is 100Hz.

¹<https://archive.ics.uci.edu/ml/datasets/PAMAP2+Physical+Activity+Monitoring>

OPP [135]: OPPORTUNITY Activity Recognition Dataset (OPP)² is got from an experimental setting simulating the breakfast scenario. The dataset comes from the recordings of 4 participants. In the process of data collecting, individuals have their freedom in the sequence and the manner of performing a set of activities. Different kinds of sensors are utilized. These sensors include the wearable sensors which are worn on the body of the participants, the ambient sensors attached on the objects (such as the sensors attached on the cup, on the spoon), and the ambient sensors which are deployed in the environment (such as the sensors deployed on the door, on the chair). The sampling frequency is 30Hz. In this dataset, the annotated labels are on five aspects: the modes of locomotion, the actions of the left the hand, the actions of the right hand, the high-level activities, and the mid-level activities.

TUD [75]: TU Darmstadt Dataset (TUD)³ is got from daily living activities of a male participant. This dataset consists of sensors readings from two wearable accelerometers positioned at the dominant wrist and in the right pocket of the trousers. It records everyday activities of him in a period of seven days. The original sampling frequency is 100Hz. For the dataset that is publicly available, the sensor readings are down sampled to 2.5Hz. The dataset is annotated on two aspects: the activities (there are 34 activity classes) and the routines (there are 4 routine classes).

3.3.2 Experimental Setup

3.3.2.1 Preprocessing

In the stage of preprocessing, we need to extract instances from the sensor readings and define the attributes and prototypes for the semantic space.

Feature extraction. For these three datasets, for each dimension of the sensor readings, we firstly fill in missing values with 0. Then, we normalize the raw sensor readings. In

²<https://archive.ics.uci.edu/ml/datasets/OPPORTUNITY+Activity+Recognition>

³<https://www.mpi-inf.mpg.de/departments/computer-vision-and-multimodal-computing/software-and-datasets/>

existing works, there are different data normalization approaches [60, 71]. Among them, the min-max normalization approach and the z-score normalization approach obtained wide adoption [88]. The z-score normalization approach is mainly suitable for cases in which the data follow normal distribution. However, through the observation of the data, we find that, a large proportion of the raw sensor readings on different dimensions do not follow normal distribution. In contrast, the min-max normalization approach does not have assumptions on the distribution of the data. Thus, we consider the min-max normalization approach is more suitable for our problem. In this work, we adopt the min-max normalization approach to perform the normalization, and the normalization is performed in this way,

$$x_{norm(i)} = \frac{x_{raw(i)} - x_{\min(i)}}{x_{\max(i)} - x_{\min(i)}},$$

where $x_{raw(i)}$ is a raw sensor reading from the i th dimension, $x_{\min(i)}$ is the minimum value of the raw sensor readings in the i th dimension in the training data, $x_{\max(i)}$ is the maximum value of the raw sensor readings in the i th dimension in the training data, and $x_{norm(i)}$ is the normalized sensor reading. Through this normalization approach, the raw sensor readings are normalized into the region of $[0, 1]$. After this, as introduced in Section 3.2.1, for each dataset, we use the sliding window strategy to separate the sensor readings into segments with fixed length. One instance is extracted from each segment, and the class label of the instance is the label with the largest duration within this segment. In the following, we introduce our approach of feature extraction for each dataset.

PAMAP2 Dataset: In our experiment, we follow the way of segmentation adopted by [131], using a sliding window of 5.12 seconds and step size of 1 second between adjacent windows. As suggested by the data provider, we discard sensor readings from 3D accelerometers with scale of $\pm 6g$ (as there is another kind of 3D accelerometer, they are more accurate, and provide the same information) and orientation readings (as they are

invalid in this dataset). From the remaining sensor readings, in each segment, we extract mean value and standard deviation from each dimension. After that, we get 27,242 labeled instances from this dataset, each having 62 dimensional features.

OPP Dataset: We use the mid-level activity class labels in our experiment, in which there are 17 activity classes. We follow the way of segmentation adopted by [70], using a sliding window of 1 second and step size of 0.5 second between adjacent windows. We discard data in the drill runs, as they are not got by individuals freely behaving, but by letting them repeat all activities in a specific sequence. From sensor readings in each segment, we extract mean value and standard deviation from each dimension. In this way, the features of each instance are of 484 dimensions. After that, we get 7,171 labeled instances from this dataset.

TUD Dataset: We use the labels of the activities in this dataset. We follow the way of segmentation adopted by [33, 34], using a sliding window of 30 seconds and step size of 15 seconds between adjacent windows. For an instance, the features come from two parts: from the sensor readings in each segment, we extract mean value and standard deviation in each dimension; we also include the time of a day as a feature. After that, features of each instance are of 13 dimensions, and we get 17,069 labeled instances from this dataset.

Attribute and prototype defining. As introduced in Section 3.2.1, we adopt the binary attribute space as the semantic space. Thus, it is necessary to define the attributes and the prototypes for each dataset. For the TUD dataset, we use the attributes and the prototypes defined in [34]. For the other two datasets, PAMAP2 and OPP, there are no defined attributes and prototypes. We define the attributes and the prototypes by ourselves.

When defining the attributes, we mainly focus on two aspects: the movement of the body; and the related object and environment. They are the two important aspects when

describing the activities. When humans perform different activities, these activities are consisted of many movements. Some activities have common movements, while others do not. For example, the activities “running” and “walking” have the common movement of “arms moving”, while activity “standing” does not have. These movements are helpful to distinguish different activities. On the other hand, for some activities, the related object and environment is important for recognizing them. For example, for the activity “playing soccer”, “soccer” is an important object for recognizing it.

When we define the attributes, we also refer to the attributes defined in [34] and [97]. In this work, we define 42 attributes for the dataset PAMAP2, and 15 attributes for the dataset OPP. After defining the attributes, the prototypes for each class are also defined. The detail of the defined attributes and prototypes is shown in Appendix A.1.

In these three datasets, there are many instances belonging to the null class. The null class refers to either not relevant activities or no activities at all [171]. In our experiment, we discard instances belonging to the null class. In the prototypes defined by [34] for the dataset TUD, there is no prototype defined for the activity “preparing food”. In our experiment, we also discard instances belonging to this class.

3.3.2.2 Comparison Methods

Our method is for zero-shot sensor-based activity recognition under the CIII setting. To evaluate the effectiveness our method, we compare our method with four baseline methods under the CIII setting.

SVMBM [34]: This method is a part of the activity recognition system NuActiv proposed in [34]. In this method, a binary classifier is learned for each attribute (each dimension of the semantic space). With the learned classifiers, instances are projected into the semantic space and 1NN classification is performed in it. This method belongs to the projection methods introduced in Section 2.2.3. This method is the representation

of state-of-the-art sensor-based zero-shot activity recognition methods. In this method, SVM is used as the attribute classifiers. Here, we refer to this method as *SVM-classifier-based method (SVMBM)*.

RegBM [114]: This method also belongs to the projection methods. Different from SVMBM, the projection is achieved by regularized linear regression. It is an implementation of the general SOC classifier proposed in [114]. This method represents the methods similar to SVMBM. In this method, as the instance projection is achieved by regression, we refer to this method as *Regression-based method (RegBM)*.

BCBM: This is the kind of *bilinear compatibility function based method (BCBM)*. It can be seen as the degeneration of our method. In existing works, there are different implementations [6, 8, 52, 139] of this kind of method. They can achieve state-of-the-art results in problems in the area of computer vision. In our experiment, except the classification model, we use the same implementation of our NCBM method. In this way, the comparison of the experimental results is more persuasive.

SynC [27]: It is a state-of-the-art zero-shot learning method belonging to the relationship methods. In this method, some phantom classes are involved. Classifiers for the unseen classes are obtained with the learned classifiers for the phantom classes and the relationships among the phantom classes and the unseen classes. The involvement of the phantom classes can be seen as an extension of the general approaches in relationship methods that rely on the classifiers for the seen classes as introduced in Section 2.2.3. This method is proposed for problem of image classification in computer vision. Here, we use it as a baseline method for our problem. In [27], three different implementations (one-versus-other (ovso), Crammer-Singer loss (cs), and Crammer-Singer with structured loss (struct)) are used to learn the classifiers. We compare with all of these three implementations.

3.3.2.3 Implementation Details

Our method is implemented by Theano [147]. The nonlinear projection function $g(\cdot)$ in Equation 3.2 is implemented as hyperbolic tangent function. In the training phase, we set the size of the mini-batch to be 100. Adagrad method [45] is adopted to update the parameters. For datasets PAMAP2 and OPP, we set the initial learning rate to be 0.01. For dataset TUD, we set the initial learning rate to be 0.5. In all of these three datasets, we set the number of training epochs to be 2000. In each dataset, when choosing the values of the hyperparameters, we use the grid search approach. In this process, we form the Cartesian product of different values of the hyperparameters, and then search for the values of them. Cross validation is utilized while performing grid search.

3.3.2.4 Evaluation Metrics

In our experiment, we use average per-class accuracy as the evaluation metric. It is the most widely used metric to evaluate zero-shot learning methods in existing works [8, 27, 129, 139, 165]. Average per-class accuracy is the average of accuracy value of each class, and calculated as

$$\text{Average per-class accuracy} = \frac{1}{N} \sum_{i=1}^N \frac{R_i}{Q_i},$$

where N is the number of classes during training or testing, R_i is the number of correctly classified instances for class i , and Q_i is the total number of instances for class i .

In sensor-based activity recognition problems, the number of instances belonging to different classes differs a lot. If we use the accuracy averaged over all testing instances as the evaluation metric, the testing result will inclines to classes with more instances. Average per-class accuracy is a metric that is not influenced by this phenomenon, and being suitable for our problem.

Table 3.1: Statistics of the number of instances and classes belonging to the seen and the unseen classes in each fold in the three datasets.

(a) Number of instances.

Fold	Number of Instances					
	PAMAP2		OPP		TUD	
	seen classes	unseen classes	seen classes	unseen classes	seen classes	unseen classes
fold 1	21461	5781	6007	1164	16423	646
fold 2	21465	5777	5617	1554	16704	365
fold 3	19995	7247	5062	2109	3619	13450
fold 4	22736	4506	6148	1023	16460	609
fold 5	23311	3931	5850	1321	15070	1999

(b) Number of classes.

Fold	Number of Classes					
	PAMAP2		OPP		TUD	
	seen classes	unseen classes	seen classes	unseen classes	seen classes	unseen classes
fold 1	14	4	13	4	26	7
fold 2	14	4	13	4	26	7
fold 3	14	4	14	3	26	7
fold 4	15	3	14	3	27	6
fold 5	15	3	14	3	27	6

3.3.2.5 Evaluation Setting Design

In our experiment, we use cross validation to evaluate the effectiveness of our method. We randomly split classes in each dataset into 5 disjoint folds, and take classes in each fold as the unseen classes in turn. Table 3.1 is the statistics of the number of instances and classes belonging to the seen and the unseen classes in each fold. Detailed information of class splitting is in Appendix A.1.

3.3.3 Evaluation on Performance

Table 3.2 shows the results of different methods on these three datasets. Each value is the average of the results in the 5 folds.

Table 3.2: Comparison on average per-class accuracy (in %) of different methods.

Method	PAMAP2	OPP	TUD
SVMBM	49.99	27.60	29.53
RegBM*	53.16	55.62	28.99
BCBM	60.61	76.31	30.19
SynC-ovso	47.24	68.62	26.01
SynC-cs	50.82	75.74	36.81
SynC-struct	54.42	78.55	31.90
NCBM**	64.37	84.10	37.35

* In this method, the results in PAMAP2 and OPP are got when the regularization parameter equals 0.1, in TUD when it equals 0.01.

** In this method, the results in PAMAP2 and TUD are got when the hidden space is of dimension 80, in OPP when the hidden space is of dimension 90.

For RegBM, we test the regularization parameter in range of $\{0.01, 0.1, 1, 10, 100\}$, and report the best results on each dataset. Both RegBM and SVMBM belong to the projection methods. For these two methods, RegBM generally outperforms SVMBM. We consider this is because in RegBM, it learns a projection function which takes all dimensions of the semantic space as a whole. The projection is optimized for all dimensions. While in SVMBM, classifiers for each dimension are learned independently. The optimization for each classifier is not for the whole projection. Thus, the performance of SVMBM is not as good as that of RegBM in general.

The performances of BCBM and SynC are comparable in general. Although on a specific dataset, one method may outperform another, but in general, their performances are not of much difference. These two methods are state-of-the-art methods in area of computer vision, for problems such as image recognition. Here we evaluate their performances in problem of sensor-based zero-shot activity recognition. The results show that, they can get acceptable results in this problem.

In all of these three datasets, our method NCBM gets the best results. In the category of compatibility based methods, it outperforms BCBM. This is in accordance with our perspective that the nonlinear classifiers are more capable than the linear classifiers. The

NCBM method also outperforms methods belonging to other categories. In methods SVMBM and RegBM, the projection functions of the instances are learned. The instances are projected into the semantic space, and 1NN classification is performed in it. In these two methods, the projection and the classification are two separate processes. In method SynC, the relationship among different classes in the semantic space is utilized to get classifiers for the unseen classes. In this method, firstly the prototypes and classifiers for the phantom classes are learned, then through the relations among the prototypes of the phantom classes and the unseen classes, classifiers for the unseen classes are constructed. Also they are two separate processes. Different from these methods, which consist of two separate processes, in our method NCBM, the model optimization is in a unified process. We consider this leads to our method outperforming these methods. The results show that the compatibility based approach is suitable for sensor-based zero-shot activity recognition problem.

3.3.4 Impact of Hidden Space Dimension

We evaluate the impact of the dimension of the hidden space. The results are shown in Figure 3.3. In the experiment, we change the hidden space dimension from 10 to 100, with an interval of 10. Figure 3.3 shows the average of the results in the 5 folds with different dimensions. From the figure we can see, when the dimension of the hidden space is extremely low (in the case of 10-dimensional), the performance on all these three datasets is relatively poor. With the increase of the hidden space dimension, the performance gets better, and finally the performance becomes stable.

When the hidden space dimension is low, the capability of the model in our method is limited. With the increase of the dimension, the capability increases. This phenomenon also illustrates the necessity of the nonlinear classifiers. The nonlinear classifiers are generally more capable than the linear classifiers. In zero-shot learning problems, they

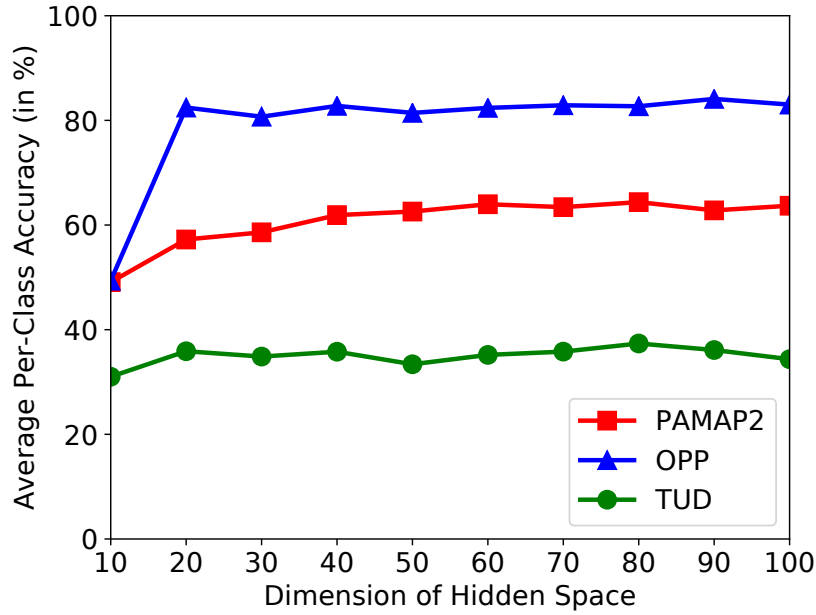


Figure 3.3: Performance of the NCBM method on the three datasets with different hidden space dimensions.

can achieve better performance. When the dimension of the hidden space is within a reasonable scope (in the scope of 50 to 100), the performance on these three datasets is relatively stable. We can conclude that, within a reasonable hidden space dimension scope, our method is not sensitive to the dimension of the hidden space.

3.3.5 Evaluation on Model Learning Time

We evaluate the time of model learning in these methods. All the experiments are conducted on a computer with Intel Core i7 CPU and 16GB memory. Table 3.3 is the statistics of the time spent by different methods when learning the model. Each value is the average of the time in the 5 folds.

In RegBM, regularized linear regression is used for instance projection. In this method, analytical solutions can be got for the model parameters. Thus, there is not an obvious process of model learning in this method. In SVMBM, it learns SVM classifiers for each dimension in the semantic space. Thus, the model learning time of it is sensitive

Table 3.3: Comparison on model learning time (in seconds) of different methods.

Method	PAMAP2	OPP	TUD
SVMBM	886.97	89.92	18.87
RegBM	-	-	-
BCBM	213.90	48.93	90.88
SynC-ovso	158.54	172.19	99.31
SynC-cs	313.95	515.82	236.55
SynC-struct	312.61	511.00	237.19
NCBM*	347.99	159.55	201.37

* In this method, the model learning time in PAMAP2 and TUD is when the hidden space of dimension 80, in OPP is when the hidden space of dimension 90.

to the dimension of the semantic space. In PAMAP2, the dimension of the semantic space is 42, much larger than the dimension of the semantic spaces in OPP (which is 15) and TUD (which is 17). Thus, in PAMAP2, SVMBM needs extreme long time for model learning. From the view of model learning time, we consider SVMBM is not suitable for high-dimensional semantic spaces. In SynC, the three implementations take different time to learn the model.

Compared with BCBM, our NCBM takes longer time for model learning. It is in accordance with our perspective. In NCBM, there is an additional projection step, and more parameters are needed to learn. In TUD, the dimensions of both the feature space and the semantic space are not high, but the learning time of our NCBM method is not very short. We consider this is because, for TUD, we set the initial learning rate to be 0.5. In our implementation, if the learning rate is set to be relatively large, the time of model learning will be relatively long.

From Table 3.3 we can see, compared with other methods, the model learning time of our method is acceptable. Also our method can get the best performance. In view of the time spend in model learning, our method is a suitable method.

3.3.6 Discussion

From the experiments, we can see the proposed NCBM method can effectively address the sensor-based zero-shot CIII-setting activity recognition problem. Existing compatibility based zero-shot learning methods are proposed and applied mainly in the area of computer vision. The experimental results in this chapter show that, the compatibility based zero-shot learning methods can be effective for problems in the area of sensor-based activity recognition. Especially, our proposed NCBM method which aims to learn nonlinear one-vs-rest classifiers can outperform the state-of-the-art methods.

For the zero-shot learning methods, the involved semantic information is important. Although the attribute space is widely used in problems in the area of computer vision, the effect of the attribute space in sensor-based activity recognition problems is just evaluated in limited works [33, 34]. For the attribute space, the defined attributes should cover the important aspects of the activities. In addition, the defined prototypes should represent the characteristics of the activities and insure the activities can be differentiated from each other. In the experiment, when defining the attributes for the datasets PAMAP2 and OPP, we focus on the movement of the body, and the related object and environment. The experimental results show the effectiveness of these defined attributes and the corresponding defined prototypes. In the future, more systematic approaches to define the attributes for different datasets can be explored. In addition, different data preprocessing approaches for the sensor readings can be explored. For example, when dividing the sensor readings, utilizing sliding windows of various lengths can be explored. For another example, when normalizing the sensor readings, novel data normalization approaches that are specific for zero-shot activity recognition problems can be explored as a future work.

3.4 Summary

In this chapter, we study the problem of unseen activity recognition for one person. It is a zero-shot learning problem under the CIII learning setting. We propose a compatibility based method NCBM to address this problem. This method learns a compatibility function between the feature space and the semantic space. To the best of our knowledge, this is the first work utilizing the compatibility function to address the sensor-based zero-shot activity recognition problem. In the experiments, the performance, the sensitivity to the hidden space dimension, and the model learning time are evaluated. The experimental results show the effectiveness of our method.

Chapter 4

Activity Recognition across Smart Homes with Different Binary Sensors and Label Spaces

In this chapter, we study the problem of activity recognition across smart homes with different binary sensors and label spaces. Activity recognition is important to many applications in smart homes. It provides information about the activities of the residents. To collect data about the activities, sensor networks consist of binary sensors are widely used. However, in different smart homes, the deployed sensor networks and the activities needed to recognize can often be different. This makes the classifier learned in one smart home environment generally cannot be used in another. In this chapter, we propose a method for smart home activity recognition with binary sensors. The classification model learned with the data in one smart home can be used in the activity recognition in another, which has different sensor networks and label spaces. Experiments on publicly available real-world datasets show the effectiveness of our method.

This chapter is organized as follows. In Section 4.1, we introduce the background of the problem. In Section 4.2, we present the proposed method. In Section 4.3, we describe the experiments. Finally, in Section 4.4, we give a summary.

4.1 Background

Smart homes aim to provide context-aware services for the residents in the home environment and improve their quality of life [11]. In smart homes, various context-aware services can be provided, such as behavioral monitoring, emergency detection, and assisted living [4]. In this area, activity recognition plays an important role, as it provides information about the activity of the residents. It is the basis of many context-aware services.

4.1.1 Binary Sensors

To perform activity recognition, data about the residents' activities are needed. To collect these data, different data collecting devices can be used, such as the cameras and the sensors [11]. In smart homes, the privacy of the residents is important. Thus, it is not appropriate to use the cameras. On the other hand, the activities needed to recognize in the smart homes are generally daily living activities, such as “having dinner”, “washing dishes”, and “brushing teeth”. These activities involve large amount of interactions of the residents with the environment. Thus, as introduced in Section 2.1.1.1, in smart homes, the ambient sensors which are deployed in the houses are widely used. In addition, as introduced in Section 2.1.1.1, in existing works on smart home activity recognition, a large number of ambient sensors belong to the binary sensors [31, 112].

Binary sensors are a kind of sensors the output of which are binary values, denoting the activation information of them. Many widely used ambient sensors belong to the binary sensors, such as the switch sensors attached to the doors, the pressure sensors attached to the seats, and the passive infrared (PIR) sensors deployed at fixed locations in the house [38]. These sensors have two states, i.e., “activated” and “inactivated”. They can report the change of the environment related with them through the activation information. For example, when a door in a house is opened, the state of the switch

sensor attached on it can be changed, when the door is closed, the state of the sensor can be changed again. The binary sensors have the advantages of easy to deploy, low cost, less intrusive, and privacy preserving [29, 31]. This makes them acceptable by the residents and widely used in smart homes. Many existing research works on smart home activity recognition are conducted based on the binary sensors [3, 89, 112, 146, 153, 155]. In these works, in a smart home environment, a set of binary sensors are deployed. These binary sensors form a *sensor network* [112, 146, 155]. The readings of these deployed binary sensors are collected as the data about the residents' activities.

4.1.2 Activity Recognition across Different Smart Homes

The activity recognition problem is a classification problem. Most of existing methods on activity recognition in smart homes address this problem by utilizing some supervised classification algorithms [3, 112, 155]. The general process is as introduced in Section 1.1.2. With the labeled training instances extracted from the labeled sensor readings, they learn the classifier used for activity recognition. In these algorithms, the feature spaces, the distribution of the instances, and the label spaces in the training and the testing datasets are required to be the same.

In practice, in different smart home environments (different houses), the house layouts are generally different. Thus, the deployed sensor networks are usually different. This often leads to different feature spaces for the instances in different houses. In addition, in different houses, the set of activities we aim to recognize can also be different. For the above reasons, in smart home activity recognition problems, the classifier learned in one house usually cannot be used in the activity recognition in another house. In addition, the labeled sensor readings collected in one house usually cannot be used for the classifier learning in another house.

For a house in which we aim to perform activity recognition, a specific classifier learned with the sensor readings collected from this house is needed. However, in practical

applications, we often need to recognize activities in different smart homes [127, 153, 154]. In a new smart home, i.e., a house having no previously collected labeled sensor readings, if we aim to perform activity recognition, we have to collect and label sufficient sensor readings in this house. The process of collecting labeled sensor readings is time consuming and expensive [127, 154]. If existing labeled sensor readings collected in other houses can be utilized to learn the needed classifier, the cost of data collecting and labeling will decrease to a large extent.

Utilizing existing data in other houses to learn the classifier for a new house is a transfer learning [37, 115] problem. In transfer learning, the knowledge contained in one learning task is transferred to another task to improve its learning.

Problem description. From the view of transfer learning, the problem we aim to address in this chapter can be described as follows. We have labeled sensor readings collected in one house (following [154], we refer to it as the *source house*). With the help of these data, we aim to learn a classification model that can classify the sensor readings in another house (following [154], we refer to it as the *target house*). The sensor networks in the source and the target houses are different. In addition, the label spaces in the source and the target houses are different, i.e., in the target house, not only the sensor readings belong to the seen classes, but also the sensor readings belong to the unseen classes are needed to recognize.

This problem is challenging, as the activity recognition is performed across both different sensor networks and different label spaces. The knowledge transfer is across both the sensor networks and the label spaces. On the other hand, this problem is of practical value, as it can often be encountered in practical applications.

Existing works. In existing works in the area of sensor-based activity recognition in smart homes, there have been efforts on addressing this problem and related problems. Firstly, as reviewed in Section 2.1.3, some methods [50, 154] are proposed for activity

recognition across smart homes with different sensor networks (in which the sensor networks in the source and the target houses are different). Secondly, as reviewed in Section 2.2.4, there are also methods, such as CDAR [180], proposed for activity recognition across different label spaces in the smart home applications (in which the activities to classify in the training and the testing datasets are different). In addition, as reviewed in Section 2.1.3, there are also methods [74, 127] proposed for activity recognition across smart homes with different sensor networks and label spaces, which is similar to the setting of our problem. However, in these methods [74, 127], during model learning, the testing instances in the target house are usually needed. In many applications in smart homes, during model learning, the testing instances are not available. This limits the application scope of these methods.

Our method. In this chapter, we propose a method to address the problem of activity recognition across smart homes with different sensor networks and label spaces. In this problem, both the sensor networks and the label spaces in the source and the target houses are different. As mentioned above, the binary sensors are widely used in smart homes. In this work, we focus on sensor networks consist of binary sensors.

Different from existing methods [74, 127] which need testing instances during model learning, in our method, during model learning, no information about the target house is needed. The information about the target house (including the information about the sensors and the activities, and the testing instances) is involved in the testing phase. Thus, compared with many existing methods, our method is more suitable for practical applications. In our method, the characteristics of binary sensors are considered. In addition, the semantic information about the sensors and the activities, along with the time information is utilized. Based on this, we refer to our method as *Binary Sensor Semantic and Time Information Method (BSST)*.

4.2 Proposed Method

In this section, we firstly give a statement of the problem. Then we introduce the overall process of our method. After that, we introduce each part of our method. Finally, we give a discussion.

4.2.1 Problem Statement

In the training phase, we have the data got from the source house. In the source house, the deployed sensor network contains N_{bss} binary sensors. Sensor readings are collected from them and are labeled. These sensor readings belong to the label space $\mathcal{C}^s$, which contains N_s classes, i.e., $\mathcal{C}^s = \{c_i^s | i = 1, \dots, N_s\}$. Following the general way in transfer learning, we refer to the label space $\mathcal{C}^s$ as the *source label space*.

In the testing phase, the sensor readings needed to classify are got from the target house. In the target house, the sensor network contains N_{bst} binary sensors. Sensor readings are collected from them. These sensor readings belong to the label space $\mathcal{C}^t$, which contains N_t classes, i.e., $\mathcal{C}^t = \{c_i^t | i = 1, \dots, N_t\}$. Similarly, following the general way in transfer learning, we refer to the label space $\mathcal{C}^t$ as the *target label space*.

The binary sensor networks in the source and the target houses are different. In addition, the source and the target label spaces are different, i.e., the source label space contains the seen classes, and the target label space contains both the seen classes and the unseen classes.

Our goal is that, with the data in the source house, we learn a classification model that can predict class labels of the sensor readings in the target house.

4.2.2 Overview of the BSST Method

The overall process of our proposed BSST method is shown in Figure 4.1.

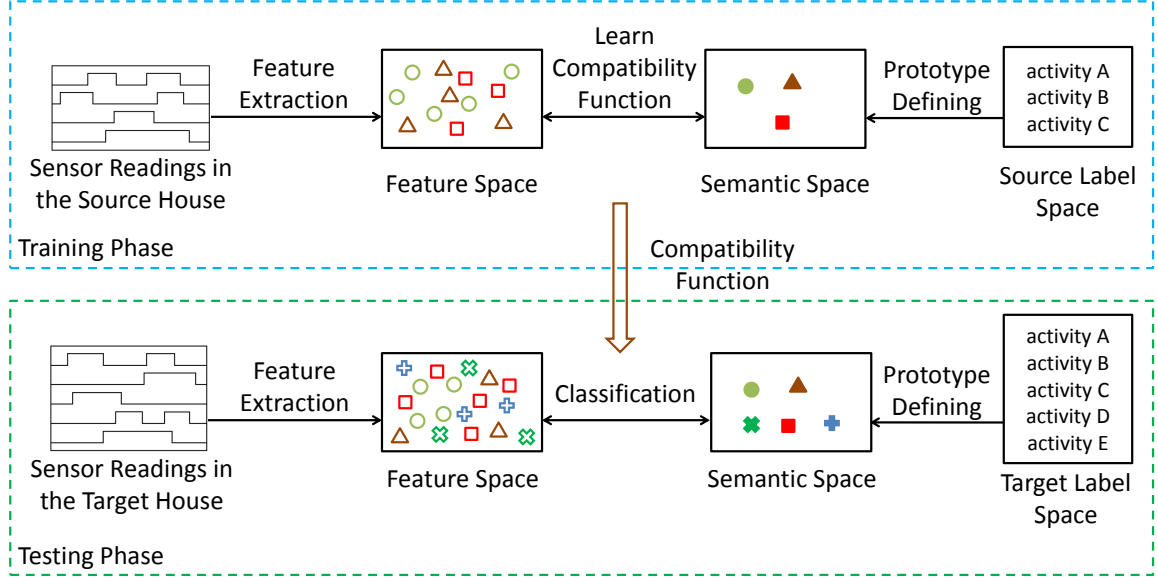


Figure 4.1: Process of the BSST method.

In our problem, the sensor networks in the source and the target houses are different. To deal with this, in the step of feature extraction, we propose an approach through which the extracted instances from different sensor networks are in a common feature space. We will introduce this step in Section 4.2.3.

In addition, in our problem, the source and the target label spaces are different. The target label space contains unseen classes which are not in the source label space. To deal with this, we follow the approach generally adopted in zero-shot learning by involving semantic information about different classes, and defining the prototypes in the semantic space. We will introduce this step in Section 4.2.4.

After extracting instances into the feature space and defining the prototypes in the semantic space, with the training instances and the prototypes of the source-label-space classes in the source house, we learn a compatibility function between the feature space and the semantic space. Then, we use it to classify the testing instances in the target house. We will introduce this step in Section 4.2.5.

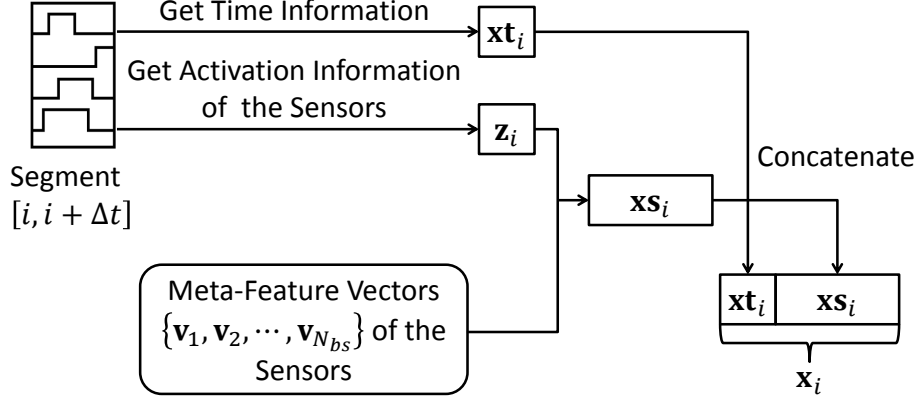


Figure 4.2: Feature extraction for a segment.

4.2.3 Feature Extraction

The sensor readings are time series data. To deal with them, we use the widely adopted sliding window strategy [3, 154, 155] to separate them into segments with fixed length. Then, we extract instances from these segments.

In existing works on smart home activity recognition, the extracted instances from different smart homes are generally in different feature spaces. For example, in [3, 155], in a smart home, when extracting instances, each dimension in the feature space corresponds to a sensor in the sensor network. In [146], in a smart home, when extracting instances, each dimension of the feature space corresponds to some statistics about the sensors in the sensor network. In different smart home, the house layout and the deployed sensors are generally different. Thus, in different smart homes, the extracted instances are generally in different feature spaces.

In our method, we propose an approach through which the instances extracted from different smart homes are in the same feature space. For a segment of the sensor readings, it contains both the time information and the activation information of the binary sensors within it. In our approach, besides the information contained in the segment itself, we also utilize the semantic information about the sensors. In addition, the characteristics

of the binary sensors are utilized. The general process of the feature extraction step is shown in Figure 4.2. For a segment, we extract instance based on the information from two parts: the time information and the sensor information (contains the activation information about the sensors and the semantic information about the sensors). In the following, we introduce these two parts.

4.2.3.1 From Time Information

In smart home activity recognition, the time information is important for the recognition of some activities. For example, the activity “having breakfast” generally occurs in the morning, and the activity “sleeping” generally occurs at night. Time information is helpful for the recognition of these activities.

To utilize the time information, we divide the time in one day into several periods: morning, noon, afternoon, evening and night. For a segment $[i, i + \Delta t]$ (where Δt is the length of the segments), we use a one-hot vector $\mathbf{x}t_i$ to represent the period it is in. In this way, in each segment, one instance is extracted from the time information. We assume the one-hot vectors are of dimension B , and denote the space of the one-hot vectors as $\mathcal{L}$.

4.2.3.2 From Sensor Information

In our problem, the sensor networks in the source and the target houses are different. When extracting instances from the sensor information, in the approaches in existing works [3, 146, 155], the extracted instances from these two houses are generally in different feature spaces. In our method, we use a different approach to extract instances. Specifically, we use the meta-features [50, 154] to construct the feature space. Instances extracted from the two different houses are in this common feature space.

Meta-features for sensors. In smart home activity recognition, the meta-features are generally a set of terms describing the properties of the sensors. In existing works, there

are different kinds of meta-features used. They can be the semantic descriptions of the sensors [154]. For example, a sensor deployed on the fridge in the kitchen can have meta-features like “located in the kitchen” and “meal related”. They can also be some statistical information about the sensors. Examples of these meta-features are “average number of activated times in an hour” and “average of the time between this sensor event and the previous sensor event” [50]. When labeled instances in the target house are available, they can also be statistics related to the class labels [50].

As introduced in Section 2.1.3, in existing works, meta-features have been used in activity recognition across smart homes with different sensor networks. In some works [35, 50, 153], they are used to get a mapping between the sensors or the feature spaces in the source and the target houses. In other works [154], they are used to provide the prior for the parameters of the classifier in the target house.

In our method, we use the meta-features to form the feature space. The extracted instances in both the source and the target houses are in this feature space. The meta-features we adopted are the semantic descriptions of the sensors. Assume we totally defined M meta-features $\{m_1, m_2, \dots, m_M\}$, these meta-features form a space $\mathcal{V}$ of dimension M . Each dimension corresponds to a meta-feature. For a sensor s_i , it has a corresponding vector $\mathbf{v}_i$ (we refer to it as the *meta-feature vector* of sensor s_i) in this space. The value $v_{i,j}$ of the j th dimension of this vector is determined by whether the sensor s_i has the corresponding meta-feature m_j . If it has this meta-feature, the value of $v_{i,j}$ is 1; otherwise, is 0. This representation of meta-feature vectors is in accordance with that in [154]. The meta-features and the meta-feature vectors of the sensors are defined manually.

We adopt the semantic descriptions as the meta-features based on the following reasons. Firstly, the semantic descriptions of the sensors can be obtained from the common sense. Thus, the meta-features and the meta-feature vectors defined in this way are not

hard to obtain. Secondly, different from the statistical information of the sensors, the semantic descriptions do not depend on the statistical characteristics of the datasets. In many applications, the sensor readings in the target house are available sequentially. There are no sufficient sensor readings when defining the meta-feature vectors for the sensors in the target house. This makes the semantic descriptions more suitable.

The meta-features we used are the semantic descriptions. They are similar to the attributes (introduced in Section 2.2.2.1) used in zero-shot learning. Each meta-feature corresponds to a property of the sensors. Accordingly, the meta-feature vectors are similar to the prototypes when adopting attribute space as the semantic space. One meta-feature vector denotes the information about whether the corresponding sensor has each of the meta-features.

Extract instances from the sensor readings. With the defined meta-features and meta-feature vectors, we extract instances into the space $\mathcal{V}$. In a sensor network (in the source house or the target house) with N_{bs} binary sensors $\{s_1, s_2, \dots, s_{N_{bs}}\}$, we have the defined meta-feature vectors $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{N_{bs}}\}$ for each of them. In the sensor network, the sensors are the binary sensors. The binary sensors just have two states: activated and inactivated. For a segment $[i, i + \Delta t]$, we use the vector $\mathbf{z}_i = (z_{i,1}, z_{i,2}, \dots, z_{i,N_{bs}})^T$ to denote the activation information of each sensor. The value of $z_{i,j}$ (which is either 0 or 1) is determined by the activation information of the corresponding sensor s_j in this time segment.

Different activation information can be used. Proposed by van Kasteren et al. [155], and widely used in existing works, there are three kinds of activation information representations used for the binary sensor networks: the raw representation, the change-point representation, and the last-sensor representation.

Raw representation: the value of $z_{i,j}$ is 1 if s_j is activated in this time segment, and 0 otherwise.

Change-point representation: the value of $z_{i,j}$ is 1 if s_j changes state in this time segment, and 0 otherwise.

Last-sensor representation: the value of $z_{i,j}$ is 1 if s_j is the sensor changes the state latest, and 0 otherwise.

With these data, the extracted instance $\mathbf{x}\mathbf{s}_i$ of this time segment is

$$\mathbf{x}\mathbf{s}_i = \sum_{j=1}^{N_{bs}} z_{i,j} \mathbf{v}_j, \quad (4.1)$$

where $\mathbf{v}_j$ is the meta-feature vector of the sensor s_j . The extracted instance $\mathbf{x}\mathbf{s}_i$ is the sum of the meta-feature vectors of sensors with activation value (i.e., $z_{i,j}$) of “1” in this segment. In this way, the extracted instances are all in this M -dimensional meta-feature space. For segments during which there are no sensors with activation value of “1”, we use zero vectors of dimension M as the extracted instances.

4.2.3.3 Concatenating Instances from Two Parts

For a segment $[i, i + \Delta t]$, through the information from the above two parts (time information and sensor information), we get two instances $\mathbf{x}\mathbf{t}_i$ and $\mathbf{x}\mathbf{s}_i$. As shown in Figure 4.2, we concatenate them to get the extracted instance $\mathbf{x}_i$ of this segment, i.e., $\mathbf{x}_i = \mathbf{x}\mathbf{t}_i \parallel \mathbf{x}\mathbf{s}_i$ (where $\parallel$ is the concatenation operator). The instances extracted in this way are all in the same feature space $\mathcal{X} = \mathcal{L} \times \mathcal{V}$, which is of dimension $D = B + M$. In this feature space $\mathcal{X}$, for the part $\mathcal{L}$ from the time information, each dimension corresponds to a time period; for the part $\mathcal{V}$ from the sensor information, each dimension corresponds to a meta-feature.

For the segment $[i, i + \Delta t]$, when extracting instance $\mathbf{x}\mathbf{s}_i$ from the sensor information, as mentioned above, different activation representations (the raw representation, the change-point representation, and the last-sensor representation) can be used as $\mathbf{z}_i$. In our method, these three activation representations are adopted and accordingly three

kinds of features are extracted. As in the extracted instances, the semantic information about the sensors and the time information is involved, we refer to the three kinds of features as *semantic-raw and time feature (SRT)*, *semantic-change-point and time feature (SCT)*, and *semantic-last-sensor and time feature (SLT)* respectively.

From the above introduction we can see, in our proposed feature extraction approach, for a segment, the contained time information and sensor-activation information is utilized. The semantic information (the meta-feature vectors) of the sensors is also utilized. When extracting instances from the semantic information and the activation information of the sensors, the characteristics of the binary sensors are utilized. In this approach, in the feature space, for the part from the time information, each dimension corresponds to a time period, for the part from the sensor information, each dimension corresponds to a meta-feature. The extracted instances from the source house and the target house are in the same feature space.

4.2.4 Prototype Defining

In our problem, the source label space and the target label space are different, i.e., $\mathcal{C}^s \neq \mathcal{C}^t$. The source label space $\mathcal{C}^s$ contains the seen classes, the target label space $\mathcal{C}^t$ contains both the seen classes and the unseen classes. To be able to recognize the classes in the target label space, we use the approach widely used in zero-shot learning [166] by defining prototypes of different classes in the semantic space.

For a class c_i in the source or the target label space, we define a corresponding prototype $\mathbf{t}_i$ in the semantic space $\mathcal{T}$. Assume the semantic space is of dimension K , after prototype defining, the class c_i has a corresponding K -dimensional prototype $\mathbf{t}_i$ in the semantic space. In our method, we adopt the binary attribute space as the semantic space. The attributes and the prototypes of the classes are defined manually. We adopt binary attribute space as the semantic space because our problem is smart home activity

recognition, the attributes and the prototypes can be defined by the common sense, and they are not hard to define.

In our problem, from the view of activity classes, the problem setting is a little different from that in zero-shot learning. In zero-shot learning, the classes (the seen classes) covered by the training data and the classes (the unseen classes) covered by the testing data are strictly disjoint. In our problem, the target label space contains unseen classes, and simultaneously it also contains the seen classes. As introduced in Section 2.2.3.3, this problem setting is referred to as generalized zero-shot learning [166] in existing works.

4.2.5 Compatibility Function

After the above two steps, the problem can be described as follows.

- In the training phase, we have N_{tr} labeled training instances $\{(\mathbf{x}_i^{tr}, y_i^{tr})\}_{i=1}^{N_{tr}}$ extracted from the sensor readings in the source house. For each labeled instance $(\mathbf{x}_i^{tr}, y_i^{tr})$, $\mathbf{x}_i^{tr} \in \mathcal{X}$ is the instance in the feature space, and $y_i^{tr} \in \mathcal{C}^s$ is its label. In addition, in the semantic space $\mathcal{T}$, we have the prototypes $\{\mathbf{t}_i^s\}_{i=1}^{N_s}$ of the N_s classes $\{c_i^s\}_{i=1}^{N_s}$ in the source label space $\mathcal{C}^s$.
- In the testing phase, we have N_{te} testing instances $\{\mathbf{x}_i^{te}\}_{i=1}^{N_{te}}$ extracted from the sensor readings in the target house. These testing instances are in the feature space $\mathcal{X}$ and belong to the target label space $\mathcal{C}^t$. They are the instances needed to classify. In addition, in the semantic space $\mathcal{T}$, we have the prototypes $\{\mathbf{t}_i^t\}_{i=1}^{N_t}$ of the N_t classes $\{c_i^t\}_{i=1}^{N_t}$ in the target label space $\mathcal{C}^t$.

Being similar to many zero-shot learning methods [6, 8, 52, 139, 166], we learn a compatibility function $F(\cdot, \cdot)$ between the feature space and the semantic space, and use it to classify the testing instances. As introduced in Section 3.2.3, it belongs to the compatibility based methods. In our problem, the compatibility function takes a feature

space instance $\mathbf{x}_i$ and a semantic space prototype $\mathbf{t}_j$ as input, and outputs a score denoting the degree of compatibility, i.e., $F(\mathbf{x}_i, \mathbf{t}_j) : \mathcal{X} \times \mathcal{T} \rightarrow \mathbb{R}$. For a class c_j (having prototype $\mathbf{t}_j$), higher compatibility score denoting more probability of $\mathbf{x}_i$ belonging to this class c_j . In our method, we use the compatibility function in form of

$$F(\mathbf{x}_i, \mathbf{t}_j) = g(\mathbf{x}_i)^T h(\mathbf{t}_j), \quad (4.2)$$

which projects the instances in the feature space and the prototypes in the semantic space into a hidden space $\mathcal{H}$. The compatibility score is the dot product of them. This form of compatibility function is also adopted in some zero-shot learning methods [5, 129]. The two projection functions $g(\cdot)$ and $h(\cdot)$ utilized in our method are nonlinear projection functions. Assume the hidden space is of dimension H , the two functions are

$$g(\mathbf{x}_i) = \delta(W_1 \mathbf{x}_i + \mathbf{b}_1), \quad (4.3)$$

and

$$h(\mathbf{t}_j) = \xi(W_2 \mathbf{t}_j + \mathbf{b}_2), \quad (4.4)$$

where $W_1 \in \mathbb{R}^{H \times D}$, $W_2 \in \mathbb{R}^{H \times K}$, $\mathbf{b}_1 \in \mathbb{R}^H$, and $\mathbf{b}_2 \in \mathbb{R}^H$ are the parameters needed to learn. $\delta(\cdot)$ and $\xi(\cdot)$ are nonlinear functions.

In the training phase, we learn the parameters in the compatibility function with the training instances $\{(\mathbf{x}_i^{tr}, y_i^{tr})\}_{i=1}^{N_{tr}}$ and the prototypes $\{\mathbf{t}_i^s\}_{i=1}^{N_s}$ of the source-label-space classes. The objective function is

$$\min \frac{1}{N_{tr}} \frac{1}{N_s} \sum_{i=1}^{N_{tr}} \sum_{j=1}^{N_s} \max\{0, 1 - \mathbb{I}_{ij} F(\mathbf{x}_i^{tr}, \mathbf{t}_j^s)\}, \quad (4.5)$$

where $\mathbb{I}_{ij}$ is an indicator that evaluates to 1 if $\mathbf{x}_i^{tr}$ and $\mathbf{t}_j^s$ are from the same class, and -1 otherwise. The loss we use is the hinge loss. We learn the parameters using stochastic gradient descent based optimization mechanism, and the learning is performed in mini-batch manner.

Algorithm 2 Training phase of the BSST method

Input: Labeled sensor readings in the source house, classes $\{c_i^s\}_{i=1}^{N_s}$ in the source label space $\mathcal{C}^s$.

Output: Compatibility function $F(\cdot, \cdot)$.

- 1: Define meta-features and meta-features vectors for the binary sensors in the source house;
 - 2: Extract the training instances $\{(\mathbf{x}_i^{tr}, y_i^{tr})\}_{i=1}^{N_{tr}}$ from the source house sensor readings;
 - 3: Define the prototypes $\{\mathbf{t}_i^s\}_{i=1}^{N_s}$ of the classes $\{c_i^s\}_{i=1}^{N_s}$ in the semantic space;
 - 4: With $\{(\mathbf{x}_i^{tr}, y_i^{tr})\}_{i=1}^{N_{tr}}$ and $\{\mathbf{t}_i^s\}_{i=1}^{N_s}$ learn the compatibility function $F(\cdot, \cdot)$ through Equations 4.2, 4.3, 4.4, and 4.5;
 - 5: **return** Compatibility function $F(\cdot, \cdot)$.
-

The training phase of the proposed BSST method is summarized in Algorithm 2.

In the testing phase, with the prototypes $\{\mathbf{t}_i^t\}_{i=1}^{N_t}$ of the classes in the target label space and the learned compatibility function $F(\cdot, \cdot)$, classification of the testing instances $\{\mathbf{x}_i^{te}\}_{i=1}^{N_{te}}$ is performed. For a testing instance $\mathbf{x}_i^{te}$, it is classified to the class with the highest compatibility score in the target label space.

4.2.6 Discussion

In our proposed BSST method, the aim is to transfer the information in the source house to the target house. In different phases, different auxiliary information is involved. In the feature extraction step in the training phase, the semantic information about the source-house sensors is involved. In the prototype defining step in the training phase, semantic information about the source-label-space classes is involved. Similarly, in the feature extraction step in the testing phase, the semantic information about the target-house sensors is involved, and in the prototype defining step in the testing phase, semantic information about the target-label-space classes is involved.

In our problem, the aim is to transfer the knowledge contained in the source house to the activity recognition task in the target house. In this problem, the knowledge source and the target should be related and not of much difference. This insures the knowledge

contained in the source house can be transferred to the target house. Specifically, in these two houses, the deployed binary sensors can be described by a common set of meta-features. Similarly, the activity classes in these two houses can be described by a common set of attributes. We use this involved information to transfer the knowledge from the source house to the target house.

In our method, the forms of the meta-feature vectors described in Section 4.2.3 and the prototypes described in Section 4.2.4 are similar. Both of them are 0-1 vectors, and are semantic descriptions about the sensors or the activities. In our method, they are utilized to implement the knowledge transfer from the source house to the target house.

The sensors we focused on in this chapter are the binary sensors. Besides the binary sensors, there are also other kinds of ambient sensors used in the smart homes, such as the sensors the output of which are real numbers. However, in existing works, the binary sensors cover a large proportion of the ambient sensors used in the smart homes. In addition, as stated in [155], the output of many other ambient sensors can be converted to be binary values. For example, for sensors the output of which are real numbers, the output can be converted to binary output by setting some threshold values [155].

4.3 Experiments

4.3.1 Datasets

In the experiment, we use the two datasets¹ [112] OrdenezA and OrdenezB obtained from UCI Machine Learning Repository [44]. These datasets are widely used in existing works [3, 49, 76] on sensor-based smart home activity recognition. Each of these two datasets contains the sensor readings generated by the binary sensor network deployed in the house of a single resident.

¹<https://archive.ics.uci.edu/ml/datasets/Activities+of+Daily+Living+%28ADLs%29+Recognition+Using+Binary+Sensors>

In the dataset OrdonezA, there are 12 binary sensors deployed in the house, collecting data of 14 days. There are 10 different activities (including the idle/other activity, which is the time with no corresponding labeled activities) in this dataset. In the dataset OrdonezB, there are also 12 binary sensors deployed in the house. Among these 12 sensors, there are 9 sensors each of which has a corresponding sensor in OrdonezA with the same type and similar location. From the view of activity recognition, we consider there are 9 common (or similar) sensors in OrdonezA and OrdonezB, and 3 unique sensors in each of them. The data collecting duration of OrdonezB is 21 days. There are 11 different activities (including the idle/other activity) in this dataset. These 11 activities include the 10 activities in OrdonezA.

The house layouts of the two datasets are different. In each dataset, different types of sensors are used, such as the passive infrared sensors, and the pressure sensors. The sensors are deployed in different locations, such as the fridge in the kitchen, and the seats in the living room. In a house, the sensors deployed in different rooms are used to collect information about activities in different rooms. For example, the sensors deployed in the living room are used to get information about activities in the living room.

4.3.2 Experimental Setup

4.3.2.1 Feature Extraction

In these two datasets, the activities are recorded in form of activity items which contain start time, end time, and activity label. In the dataset OrdonezA, there are three activity items², in which the end time is before the start time, which is unreasonable. When preprocessing the data, we discard these three activity items. For the sensor readings in these two datasets, adopting the sliding window strategy, we separate them into segments

²The three activity items are 2011-12-01 19:28:51, 2011-12-01 16:29:59, Toileting; 2011-12-02 12:20:41, 2011-12-01 10:20:59, Grooming; 2011-12-02 12:27:47, 2011-12-01 11:35:49, Breakfast

with fixed length. We follow the way adopted in [3, 112] by setting the length of the segments to be 60 seconds, with no overlap between adjacent segments.

Then, we extract instances from these segments. As mentioned in Section 4.2.3, each instance is extracted based on the information from two parts: time information and sensor information.

- For the time information, we need to determine the different time periods in one day. We divide the time in one day into 5 periods, and set 6:00 to 11:00 being the morning, 11:00 to 13:00 being the noon, 13:00 to 18:00 being the afternoon, 18:00 to 23:00 being the evening, and 23:00 to 6:00 being the night. Each segment belongs to a time period. We use a one-hot vector with five digits to represent it.
- For the sensor information, the meta-features and the meta-feature vectors are needed to define. When defining them, we refer to the approaches adopted in [35, 153, 154]. We define 9 meta-features focusing on two aspects: location and function. The location denotes the general location in which the sensor is deployed; the function denotes according to the location in which the sensor is deployed, what function it is mainly related with. They are two important aspects to describe the sensors in the activity recognition problems. The details of the defined meta-features and the meta-feature vectors of the sensors are in Appendix A.2.

Then, for each segment, following the feature extraction approach described in Section 4.2.3, we extract the three features: SRT, SCT, and SLT. The activity label of each instance is set to be the label with the largest duration within the corresponding segment. If there are concurrent activities during a segment, we set the class label to be the activity occurs latter. After this process, we get 20,160 instances in dataset OrdonezA and 30,240 instances in dataset OrdonezB.

4.3.2.2 Prototype Defining

For the activities, we define 17 attributes on three aspects: property, time, and location. The property denotes what the activity is mainly related with; the time denotes at what time the activity generally occurs; the location denotes the general occurring location of the activity. They are three important aspects to describe the smart home activities. When defining the attributes, we refer to the way of denoting activities in [127]. The details of the defined attributes and the prototypes of the activities are in Appendix A.2.

4.3.2.3 Evaluation Metrics

In smart home activity recognition, because of the nature of the data, the classes are highly imbalanced. However, when performing activity recognition, not only the activity classes with large number of instances are needed to recognize, but also the activity classes with rare instances are also of interest. To evaluate the effectiveness of our method, we use

$$Precision = \frac{1}{N} \sum_{i=1}^N \frac{TP_i}{TP_i + FP_i},$$

$$Recall = \frac{1}{N} \sum_{i=1}^N \frac{TP_i}{TP_i + FN_i},$$

and their harmonic mean

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall},$$

as the evaluation metrics. In these evaluation metrics, N is the number of classes during training or testing, TP_i , FP_i , and FN_i are the number of true positive, false positive, and false negative instances for class i respectively. These evaluation metrics are not influenced by the distribution of different classes and are widely used in smart home activity recognition.

4.3.3 Performance Evaluation

4.3.3.1 Evaluation Design

We evaluate the performance of our proposed method through the transferring between OrdenezA and OrdenezB. We design the evaluation with the aim of to be consistent with the practical applications. To evaluate our method, we need both the sensor networks and the label spaces in the source and the target houses to be different. For the sensor networks in these two datasets, there are 9 common sensors, and 3 unique sensors for each of them. We consider this is in accordance with the practical applications. In practice, for the source and the target houses, there are some common sensors, and some unique sensors in each of them. For the class labels, there are 10 activity classes in OrdenezA and 11 activity classes in OrdenezB (one more class than OrdenezA). In the evaluation, we need the testing instances contain some unseen classes. Thus, we design the evaluation in the following way.

Firstly, we evaluate the transfer from OrdenezA to OrdenezB. In this evaluation, the training dataset is OrdenezA, and the testing dataset is OrdenezB. For the OrdenezA dataset, we retain the class labels of instances belonging to activities “leave house”, “toileting”, “taking a shower”, “sleeping”, “preparing breakfast”, “preparing lunch”, “spare time/TV”, and “idle/other”, and relabel the instances belonging to activities “preparing snack” and “grooming” to have the class label “idle/other”. This way of dealing with the data is in accordance with the practical applications. In practice, when we label the instances, just instances belonging to the predefined activity classes that are of interest are labeled, and all others are labeled as “idle/other”. We select these eight activity classes to be retained in the training data because according to the common sense, they are more necessary for daily living, and more likely to be labeled; also they are contained in both datasets. In this way, there are labeled instances of 8 activities in the training dataset, and 11 activities in the testing dataset. Then, we evaluate the transfer

Table 4.1: Transfer between OrdonezA and OrdonezB (in %).

Transfer Direction	Feature	Precision	Recall	F1-score
OrdonezA $\rightarrow$ OrdonezB	SRT	40.48	44.11	42.21
	SCT	39.29	22.42	28.55
	SLT	41.12	51.09	45.56
	Random	9.09	9.09	9.09
OrdonezB $\rightarrow$ OrdonezA	SRT	50.84	40.86	45.31
	SCT	32.85	22.80	26.92
	SLT	45.76	52.11	48.73
	Random	10	10	10

from OrdonezB to OrdonezA. In this evaluation, the training dataset is OrdonezB, and the testing dataset is OrdonezA. Similarly, the labels of instances belonging to the above eight activity classes are retained, and instances belonging to the other three activity classes are relabeled as “idle/other”. In this way, there are labeled instances of 8 activities in the training dataset, and 10 activities in the testing dataset.

4.3.3.2 Implementation Details

We use the hyperbolic tangent function as the nonlinear function $\delta(\cdot)$ in Equation 4.3 and $\xi(\cdot)$ in Equation 4.4. Adam [84] is adopted for optimization. During model learning, we set the initial learning rate η to be 0.001, the number of training epochs to be 500, and the batch size of the training instances to be 100. The dimension H of the hidden space is determined through five-fold cross validation. We implement our method using Theano [147].

4.3.3.3 Results and Analysis

The evaluation results are shown in Table 4.1. From the results, we can see the effectiveness of our method. For all of the three features and the two transfer directions, the results of our method are significantly better than that of random classification.

The results of features SRT and SLT are comparative, and better than that of the feature SCT. This difference among the results of the SRT, SCT, and SLT features is in

accordance with the results in [112]. In [112] (in which the problem setting is supervised classification), for the two datasets OrdonezA and OrdonezB, the evaluation results of the raw and the last-sensor representations are comparative, and better than that of the change-point representation. The problem setting of our method is different from that in [112], thus we cannot compare the results in these two works directly. However, the datasets used are similar, thus the results in [112] are helpful for analyzing the results of our method. From the results, we consider the difference among the results of SRT, SCT, and SLT is because of the characteristics of the datasets.

The results also show that, in our method, the process of information transferring can effectively maintain the characteristics of the datasets. The information contained in the training dataset can be effectively transferred to the task of testing instance classification.

4.3.4 Evaluation on the Effect of Amount of Training Data

We further evaluate the effect of amount of training data when transferring between the datasets OrdonezA and OrdonezB. We vary the amount of training data through varying the number of days from which to get the training instances. When instances in n days are used as training instances, we randomly select the instances in n days from the training dataset, and use them to learn the classification model. We repeat this process ten times and report the average results.

The three features and the two transfer directions between OrdonezA and OrdonezB are evaluated. The evaluation results are shown in Figure 4.3. From this figure, we can see that, for the three features, with different number of training days, the SRT and SLT features constantly outperform the SCT feature in both of the two transfer directions. This shows that, the characteristics of the datasets are maintained with different number of training days.

We can also see that, when the number of training days is less than six days, the performance roughly increases with the increase of the training days. This shows that,

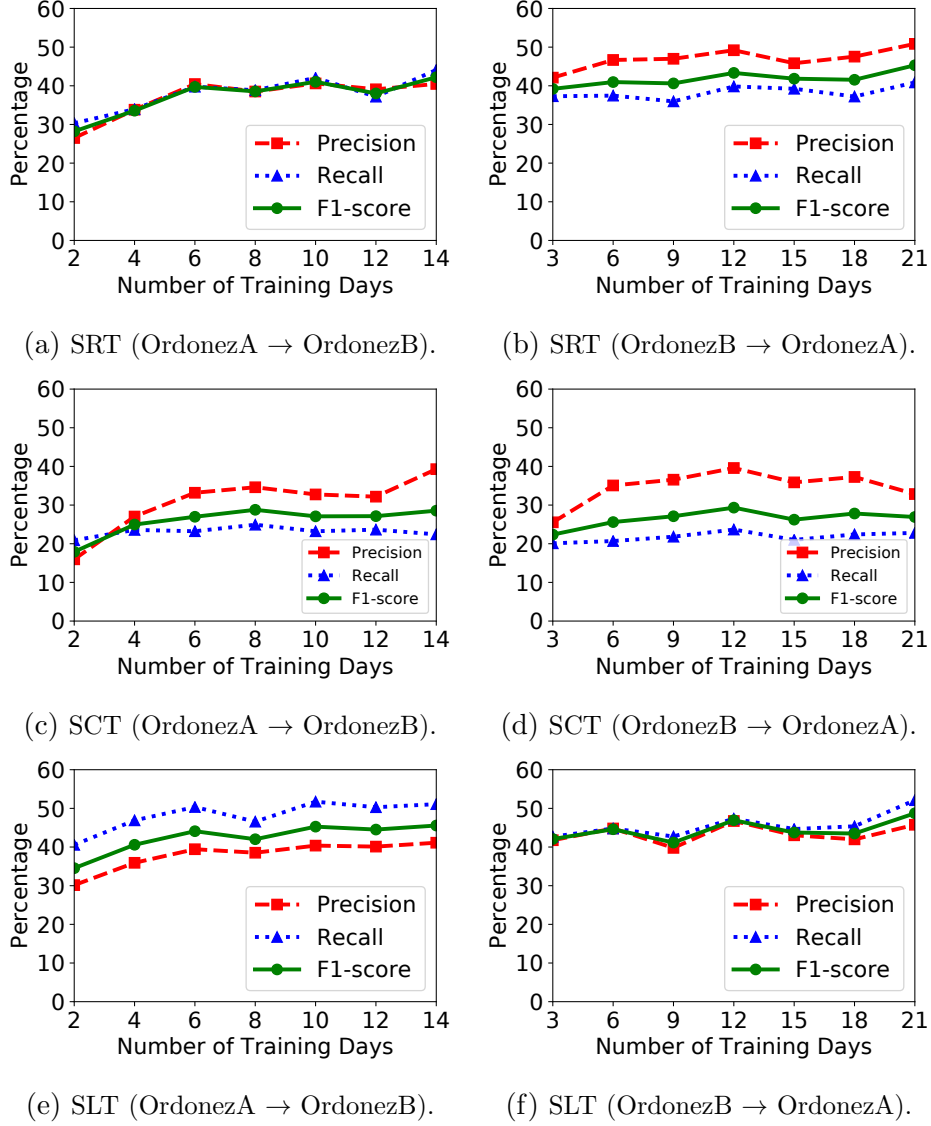


Figure 4.3: Effect of amount of training data.

when learning the classification model, the number of training instances cannot be very small. On the other hand, when the number of training days exceeds six days, the performance begins to get relatively stable. With the increase of the training days, they just fluctuate. We consider this is because in the process of information transferring, information loss is unavoidable. Some useful information in the training dataset is not effectively transferred to the task of testing instance classification. Thus, even with

the increase of the number of the training days, it does not effectively improve the classification performance of the testing instances.

4.3.5 Discussion

The results in the experiment show the effectiveness of our BSST method. The evaluation on the effect of amount of training data shows that, in our method, when the training instances increase to certain amount, the recognition results of our method do not improve with the increase of the amount of training instances. We consider that, in the process of information transfer, some of the useful information is not effectively utilized. As a future work, we can explore how to effectively utilize the information contained in the source house to improve the recognition results. Besides, the types and the deployed locations of the sensors in the source and the target houses are also important. In different locations, different types of sensors can capture different kinds of information. In the future work, we can explore how to select different types of sensors and deploy them in different locations, to let the activity recognition be more efficient.

In the experiment, we focus on the most common situation, in which there are some common sensors for the sensor networks in the source and the target houses, and some unique sensors for each of them. As a future work, we can explore the effect of our method in situations in which the sensors in the source and the target houses are totally different, and evaluate the effect of the percentage of common (or similar) sensors on the results.

In this work, we focus on the most common setting in which the deployed sensors in the source and the target houses are fixed. In some applications, the sensor network deployed in a smart home environment can change dynamically, i.e., new sensors can be added, and some sensors can be out of work. In our method, in these situations, the change of the sensors will not cause the change of the feature space. As in the feature

space, for the part containing sensor information, each dimension corresponds to a meta-feature. Thus, the change of the sensors will not cause the change of the feature space. However, we also need to deal with the problem caused by the changing of the sensors. In the cases of new sensors added to the sensor network, we can define meta-feature vectors for them, and incorporate them in the process of activity recognition. On the other hand, in the cases of some sensors being out of work, the out-of-working of some sensors may cause the loss of the required information. Thus, when utilizing our method in practical applications, to let it work efficiently, we need some checking mechanisms to check whether the sensors are working properly. If there exists sensor detected being out of work, we need to replace it. In the future work, we can explore methods for situations in which the deployed sensors can change dynamically. On the one hand, when there are new sensors added to the sensor network, how to more efficiently utilize them. On the other hand, when there exist sensors being out of work, how to efficiently detect and report this, for timely sensor replacement.

As another future work, the deep learning approaches can be considered. In recent years, deep learning has achieved success in various areas. In deep learning approaches, during model learning, usually large amount of training instants are needed. However, in our problem, the amount of training data is relatively not large. This makes the deep learning approaches not very suitable. In the future, we will explore the potential of deep learning approaches in addressing our problem. We will focus on how to develop approaches that do not need large amount of training data during model learning.

4.4 Summary

In this chapter, we study the problem of activity recognition across smart homes with different binary sensors and label spaces. To address this problem, we propose a method BSST. In this method, the sensor readings from different sensor networks are represented

in a common feature space, and the class labels in different label spaces are represented in a common semantic space. A compatibility function between these two spaces is learned. In our method, the semantic information of the sensors and the activities, the time information about the sensor readings, and the characteristics of the binary sensors are utilized. To the best of our knowledge, this is the first work for activity recognition across smart homes with different sensors networks and label spaces, in which during model learning, no information about the target house is needed. Our method focuses on the sensor networks consist of binary sensors. In the experiments, the performance of the method, and the effect of amount of training data are evaluated. The experimental results show that our method is effective.

Chapter 5

Multi-Resident Activity Recognition Including Unseen Classes in Smart Homes

In this chapter, we study the problem of multi-resident activity recognition including unseen classes in smart homes. Multi-resident activity recognition is important to many applications in smart homes. Existing works in this area are generally based on supervised classification algorithms, and can only recognize activities that have labeled instances in the training data. However, in many applications, not only these activities, but also some previously unseen activities are needed to recognize. We propose a method to address the problem of multi-resident activity recognition including unseen classes. In our method, the multi-task learning and the zero-shot learning techniques are utilized. The activity recognition of each resident is regarded as a learning task, and all tasks are learned jointly. As zero-shot learning techniques are utilized, the unseen classes can be recognized. Experiments show the effectiveness of our method.

This chapter is organized as follows. In Section 5.1, we introduce the background of the problem. In Section 5.2, we present the method proposed to address the problem. In Section 5.3, we describe the experiments. Finally, in Section 5.4, we give a summary.

5.1 Background

Smart homes aim to assist and facilitate the living of the residents through providing context-aware services [11]. To provide these services, being aware of the residents' activities is needed, as many services are based on the activities performed by the residents. To collect information about the activities, as stated in Section 2.1.1.1, ambient sensors deployed in the environment get wide adoption. These sensors are generally considered as less intrusive [155] and more privacy preserving [146]. In addition, information about the environment can be captured by them. This makes them suitable for activity recognition in smart homes, in which the activities needed to recognize are generally daily living activities containing many interactions with the environment.

5.1.1 Multi-Resident Activity Recognition in Smart Homes

Multi-resident activity recognition. There have been extensive amount of research works on smart home activity recognition based on the ambient sensors [3, 89, 112, 146, 155]. However, these works mainly focus on the single-resident setting. Under the single-resident setting, there is only one resident in a smart home environment. Thus, the obtained sensors readings are solely about this resident. However, in many practical scenarios, the residents do not live along, and there are multiple residents in a smart home environment.

Performing activity recognition under the multi-resident setting is more challenging than under the single-resident setting. As the ambient sensors are deployed in the environment, when there are multiple residents, the sensor readings about different residents are mixed together. In addition, in a smart home environment of multiple residents, the activities of one resident can be influenced by other residents. This makes the activity recognition under the multi-resident setting being more complex.

Due to these reasons, compared with the research works on single-resident activity recognition, the number of research works on multi-resident activity recognition is much smaller. In recent years, it obtains increasing interest in the area of smart home activity recognition and gets some development [19].

Unseen activities. In existing works on multi-resident activity recognition, the general process follows the process shown in Section 1.1.2. In these works, in the training phase, to learn the classifier, adequate sensor readings belonging to a set of predefined activity classes are collected and labeled. Training instances are extracted from them. With these instances, the classifier is learned. However, being similar as in the single-resident setting [155], the learned classifier can only recognize these predefined activity classes.

In many practical applications, the recognition scenario and the recognition aim may change over time. In this case, when we use the learned classifier to recognize activities in the testing phase, not only the predefined activities, but also some previously unseen activities may also be needed to recognize. However, as introduced in Section 1.1.3, the process of obtaining labeled sensor readings is costly [34]. Under the setting of multi-resident activity recognition, this phenomenon is more severe, as under this problem setting, activities of more than one resident are needed to label. Thus, when there are unseen activities needed to recognize, it is not a good manner to recollect sensor readings and retrain a new classifier. Methods that both can recognize instances belonging to the predefined activities and can recognize instances belonging to the previously unseen activities are needed.

5.1.2 Problem and Existing Works

Problem description. In this chapter, we propose to address the problem of multi-resident activity recognition including the unseen classes in smart homes. As stated above, this problem can occur in many practical applications. Thus, it is a problem of

practical value. On the other hand, this problem is also challenging. As in this problem, not only activities coming from multiple residents are needed to recognize, but also these activities contain the unseen activities.

Existing works. The existing works on multi-resident activity recognition in smart homes [19] are reviewed in Section 2.1.4. As reviewed in that section, most of these existing works utilize the ambient sensors as the data collecting devices. Various recognition methods have been proposed. However, there are no works for the problem in which the activities needed to recognize including the unseen activities. On the other hand, the existing works on sensor-based unseen activity recognition (zero-shot activity recognition) are reviewed in Section 2.2.4. As reviewed in that section, these works generally focus on the setting of one person [34, 180], not for the case of multiple residents in the smart home environment. In addition, the multi-resident activity recognition is a multi-label learning problem. As reviewed in Section 2.2.3.3, existing works [54, 102, 178] for multi-label zero-shot learning are not for the problem of sensor-based activity recognition.

Our method. In this chapter, we propose a method for the problem of multi-resident activity recognition including unseen classes in smart homes. Based on the problem setting, we refer to our method as *Multi-Resident with Unseen-Activity-Class Recognition Method (MRUA)*. In our MRUA method, the techniques of multi-task learning and zero-shot learning are utilized. By regarding the activity recognition of each resident as a learning task, all tasks are learned jointly. With the zero-shot learning techniques, the unseen classes can be recognized.

5.2 Proposed Method

In this section, we introduce our proposed MRUA method. As introduced in Section 2.1.4, most of existing works in the area of multi-resident activity recognition in smart homes are based on the ambient sensors. In our work, we also assume the ambient sensors

are used as the data collecting devices. In addition, like existing works introduced in Section 2.1.4, we introduce our method in the situation of two residents, i.e., there are two residents in the smart home. Our method is also applicable to situations of more than two residents. And this is discussed in Section 5.2.4.

5.2.1 Problem Statement

In our problem, the activity classes covered by the training and the testing data are different. From the view of transfer learning [115], the aim of our method is to transfer the information contained in the training data to the task of testing data classification. Following the general way in transfer learning, being similar as in Chapter 4, we refer to the class label space of the training data as the *source label space*, and the class label space of the testing data as the *target label space*.

In the training phase, we have the sensor readings got from the ambient sensors deployed in a smart home environment of two residents. We refer to these two residents as *Resident1* and *Resident2*. The activities of these two residents are labeled. These labels belong to the source label space $\mathcal{C}^s = \{c_i^s | i = 1, \dots, N_s\}$. In the testing phase, we have the unlabeled sensor readings also got from these ambient sensors, and these sensor readings are also about these two residents. These sensor readings belong to the target label space $\mathcal{C}^t = \{c_i^t | i = 1, \dots, N_t\}$. The source and the target label spaces are different, i.e., $\mathcal{C}^s \neq \mathcal{C}^t$. The source label space contains the seen classes, the target label space contains both the seen and the unseen classes. In the training phase, with the labeled sensor readings, we aim to learn a classification model that can classify the unlabeled sensor readings in the testing phase.

5.2.2 Data Preprocessing

Feature extraction. The raw data are the sensor readings. They are time series data. Following the general way of dealing with the sensor readings [155], we adopt the sliding

window strategy to separate the sensor readings into segments with fixed length. The classification is based on the segments.

As introduced in Section 2.1.4, in existing works on multi-resident activity recognition, there are also many works performing the classification based on the sensor events [30, 73]. However, in our method, like in the works [12, 122], we choose to perform classification based on the time segments. This is because, in practical applications, we often need to get aware of the residents' activities not necessarily at the time when there are sensor events occur, but also at the time when there are no sensor events. After data segmentation, for each segment, one instance in the feature space is extracted.

Prototype defining. In our problem, the target label space contains the unseen classes. To be able to recognize the activity classes covered by the testing data, following the general way adopted in zero-shot learning, we involve the semantic information about different classes and define prototypes for them. For a class c_i in the source or the target label space, we define a prototype $\mathbf{t}_i$ in the semantic space $\mathcal{T}$.

From the view of activity classes, being similar as the problem setting in Chapter 4, this problem setting is a little different from the setting of zero-shot learning. In zero-shot learning, the source and the target label spaces are disjoint. In our problem, the source label space contains the seen classes, and the target label space contains not only the unseen classes but also the seen classes. This problem setting is referred to as generalized zero-shot learning [166] in existing works.

5.2.3 Proposed MRUA Method

Problem formulation. After the above data preprocessing steps, we formulate the problem as follows.

- In the training phase, we have N_{tr} labeled training instances $\{(\mathbf{x}_i^{tr}, y_i^{tr(1)}, y_i^{tr(2)})\}_{i=1}^{N_{tr}}$ got from the labeled sensor readings. For each labeled instance $(\mathbf{x}_i^{tr}, y_i^{tr(1)}, y_i^{tr(2)})$, $\mathbf{x}_i^{tr}$ is the

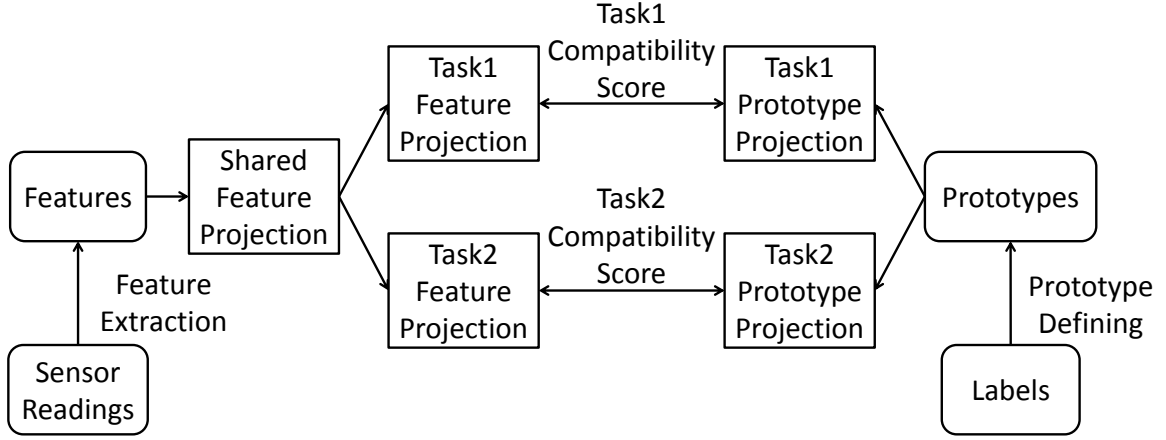


Figure 5.1: Process of the MRUA method.

instance in the feature space $\mathcal{X}$, i.e., $\mathbf{x}_i^{tr} \in \mathcal{X}$; $y_i^{tr(1)} \in \mathcal{C}^s$ is the activity label of Resident1; $y_i^{tr(2)} \in \mathcal{C}^s$ is the activity label of Resident2. The feature space $\mathcal{X}$ is a D -dimensional real number space, i.e., $\mathbb{R}^D$. In the semantic space $\mathcal{T}$, we have the prototypes $\{\mathbf{t}_i^s\}_{i=1}^{N_s}$ of the classes $\{c_i^s\}_{i=1}^{N_s}$ in the source label space $\mathcal{C}^s$. The semantic space $\mathcal{T}$ is an M -dimensional real number space, i.e., $\mathbb{R}^M$. With these data, we need to learn a classification model which can perform classification in the testing phase.

- In the testing phase, we have N_{te} unlabeled testing instances $\{\mathbf{x}_i^{te}\}_{i=1}^{N_{te}}$ in the feature space $\mathcal{X}$. In the semantic space $\mathcal{T}$, we have the prototypes $\{\mathbf{t}_i^t\}_{i=1}^{N_t}$ of the classes $\{c_i^t\}_{i=1}^{N_t}$ in the target label space $\mathcal{C}^t$. For each testing instance $\mathbf{x}_i^{te}$, we need to predict the activity label $y_i^{te(1)}$ of Resident1 and the activity label $y_i^{te(2)}$ of Resident2, both of which belong to the target label space $\mathcal{C}^t$, i.e., $y_i^{te(1)} \in \mathcal{C}^t$ and $y_i^{te(2)} \in \mathcal{C}^t$.

Multi-task learning. In our problem, for each instance $\mathbf{x}_i$, there are two corresponding activity labels $y_i^{(1)}$ and $y_i^{(2)}$ for the two residents respectively. It is a multi-label classification problem [177]. However, the setting of our problem is a little different from the general multi-label classification problem settings as defined in [177]. In the general multi-label classification problems, for an instance $\mathbf{x}_i$, the number of related labels is uncertain. The related labels can be any number of the labels in the label space. In our

problem, for each instance $\mathbf{x}_i$, the number of labels is fixed: $y_i^{(1)}$ is the label of Resident1 and is one label from the label space; similarly, $y_i^{(2)}$ is the label of Resident2 and is also one label from the label space.

In view of this, we propose to address this problem from the view of multi-task learning. Multi-task learning [26] is a learning paradigm in which there are multiple learning tasks. By jointly learning these tasks and exploiting the relations among them, multi-task learning aims to improve the learning of all these tasks. Under our problem setting, the activity recognition of each resident can be regarded as a task. Our goal is the activity recognition for both of the two residents.

In the area of sensor-based activity recognition, there have been works solving the multi-label classification problems through multi-task learning. Such as in [152], the problem setting is one-person activity recognition, in which the person can perform more than one activity at the same time. The recognition of each activity is regarded as a task, and all tasks are learned jointly. In [117], two recognition tasks, i.e., the recognition of simple activities and the recognition of complex activities, are learned jointly through multi-task learning. However, to the best of our knowledge, we have not seen works solving the multi-resident activity recognition problems from the view of multi-task learning, especially under the setting with unseen classes.

Zero-shot learning. The target label space contains the unseen classes. To be able to recognize the classes in the target label space, zero-shot learning techniques are needed. In this work, like many zero-shot learning methods [6, 8, 52, 139] belonging to the compatibility based methods, we learn a compatibility function $F(\cdot, \cdot)$ between feature space instances and the semantic space prototypes. This function takes an instance $\mathbf{x}_i$ and a prototype $\mathbf{t}_j$ as input, and outputs a score denoting the degree of compatibility, i.e., $F(\mathbf{x}_i, \mathbf{t}_j) : \mathcal{X} \times \mathcal{T} \rightarrow \mathbb{R}$. For a testing instance $\mathbf{x}_i^{te}$, it is classified to the class in the target label space $\mathcal{C}^t$ with the highest compatibility score.

Classification model. The overall process of our MRUA method is shown in Figure 5.1. In our method, activity recognition of each resident is regarded as a task, and the recognition is performed through the compatibility function. For task r ($r = 1, 2$), the compatibility function is in form of

$$F_r(\mathbf{x}_i, \mathbf{t}_j) = \delta_r(\mathbf{x}_i)^T \varphi_r(\mathbf{t}_j). \quad (5.1)$$

In this function, the instance $\mathbf{x}_i$ and the prototype $\mathbf{t}_j$ are projected into a hidden space $\mathcal{H}$ through the projection functions $\delta_r(\cdot)$ and $\varphi_r(\cdot)$ respectively, and their dot product is used as the compatibility score. In our method, to avoid the model to be too complex, we use the same hidden space $\mathcal{H}$ for the two tasks.

Projection of instances. The projection function $\delta_r(\cdot)$ for the instances is in form of

$$\delta_r(\mathbf{x}_i) = \xi(W_r^{tf} \phi(W^{sf} \mathbf{x}_i + \mathbf{b}^{sf}) + \mathbf{b}_r^{tf}), \quad (5.2)$$

which consists of functions $\phi(\cdot)$ and $\xi(\cdot)$ for two projections. The first projection is shared by the two recognition tasks. We refer to this projection as the *shared feature projection*. $\phi(\cdot)$ is the nonlinear function of this projection. Assume the output of this function is of dimension K . In this function, $W^{sf} \in \mathbb{R}^{K \times D}$ and $\mathbf{b}^{sf} \in \mathbb{R}^K$ are the parameters, and shared by the two tasks. After this, is the second projection. For this projection, each task has its own projection function. For task r , we refer to the corresponding projection as *task r feature projection*. Assume the hidden space $\mathcal{H}$ is of dimension H , in the nonlinear function $\xi(\cdot)$ of this projection, $W_r^{tf} \in \mathbb{R}^{H \times K}$ and $\mathbf{b}_r^{tf} \in \mathbb{R}^H$ are the parameters, and specific for the recognition task r .

Projection of prototypes. The projection of the prototypes is specific for each task. For task r , we refer to this projection as *task r prototype projection*. The projection function $\varphi_r(\cdot)$ is in form of

$$\varphi_r(\mathbf{t}_j) = \theta(W_r^{tp} \mathbf{t}_j + \mathbf{b}_r^{tp}), \quad (5.3)$$

in the nonlinear function $\theta(\cdot)$ of this projection, $W_r^{tp} \in \mathbb{R}^{H \times M}$ and $\mathbf{b}_r^{tp} \in \mathbb{R}^H$ are the parameters, and specific for the recognition task r .

Model learning. In our method, the model parts of task1 and task2 are learned jointly. To learn the whole model of the two residents, with the training instances and the source label space class prototypes, we optimize the objective function in form of

$$\min \frac{1}{R} \sum_{r=1}^R \left(\frac{1}{N_{tr}} \frac{1}{N_s} \sum_{i=1}^{N_{tr}} \sum_{j=1}^{N_s} \ell_r(\mathbf{x}_i^{tr}, \mathbf{t}_j^s) \right), \quad (5.4)$$

where $R = 2$ is the number of tasks (the number of residents), ℓ_r is the loss of classifying instance $\mathbf{x}_i^{tr}$ to class c_j^s (the class of $\mathbf{t}_j^s$) for recognition task r . In our method, we use the hinge loss as the loss function ℓ_r , which is in form of

$$\ell_r(\mathbf{x}_i^{tr}, \mathbf{t}_j^s) = \max\{0, 1 - \mathbb{I}_{rij} F_r(\mathbf{x}_i^{tr}, \mathbf{t}_j^s)\}, \quad (5.5)$$

where $\mathbb{I}_{rij}$ is an indicator. For task r , if the class of $\mathbf{x}_i^{tr}$ (i.e., $y_i^{tr(r)}$) and the class of $\mathbf{t}_j^s$ (i.e., c_j^s) are the same, it evaluates to 1; otherwise it evaluates to -1 . To learn this model, we use the stochastic gradient descent based method, and perform the learning in mini-batch manner. The learning process is performed to a fixed number of epochs.

5.2.4 Discussion

In our method, the combination of zero-shot learning and multi-task learning techniques is utilized to address the problem.

From the view of learning task. In our problem, for each instance in the feature space, it has the information about the activities of the two residents. In our method, the shared feature projection is firstly applied to the instances. Then, for each task r , the task r feature projection is applied to the instances, and the task r prototype projection is applied to the prototype of the class label. In our method, the shared feature projection for the instances is the shared part of the two tasks. For each task r , the task r feature

projection and the task r prototype projection are the parts specific for this task. This model structure is a widely used structure in multi-task learning. It is referred to as “hard parameter sharing” [141] or “explicit parameter sharing” [61]. In this model structure, different learning tasks have some shared parameters. In our model, through jointly learning the two recognition tasks, the performances of the two tasks are improved.

From the above illustration we can see, our method is also applicable to the situations of more than two residents. In our method, the activity recognition of each resident is a learning task. By adding more learning tasks to the model, the model can be adapted to the situations of more than two residents.

From the view of the information involved. In the training phase, besides the training instances, the semantic information about the source-label-space classes (the prototypes of the seen classes) is involved in model learning. In the testing phase, the learned classification model and the semantic information about the target-label-space classes (the prototypes of the seen and the unseen classes) are involved in classifying the testing instances.

5.3 Experiments

5.3.1 Datasets

In the experiment, we use the publicly available ARAS datasets¹ [13]. We choose these datasets because they are collected in the real world settings, and they are widely used in existing works [12, 48] on multi-resident activity recognition.

ARAS datasets consist of two datasets collected from two houses: HouseA and HouseB. In HouseA, 20 binary ambient sensors are deployed in the house, collecting data of 30 days. Two male residents live in the house. The sensor readings and the

¹<https://www.cmpe.boun.edu.tr/aras/>

activity labels of the two residents are recorded per second. There are 27 different activities labeled. In HouseB, also 20 binary ambient sensors (but not the same sensors as in HouseA) are deployed in the house. The duration of data collection is also 30 days. The residents in this house are a married couple. The sensor readings and the activity labels of the two residents are also recorded per second. The same 27 activities as in the HouseA dataset are labeled.

In these two datasets, different kinds of sensors (such as the pressure sensors and the contact sensors) are deployed in different locations in the houses (such as the doors, the chairs, and the wardrobes). In each house, information about the activities of the two residents is recorded by the deployed ambient sensors.

5.3.2 Experimental Setup

5.3.2.1 Preprocessing

Feature extraction. We preprocess the raw data in the HouseA and the HouseB datasets following the way adopted in [12]. Adopting the sliding window strategy, we separate the sensor readings into fixed-length segments. The length of the sliding window is set to be 60 seconds. There are no overlaps between adjacent segments. For each segment, an instance is extracted from the sensor readings. In each dataset, as there are 20 sensors, the extracted instance is a vector of dimension 20, with each dimension corresponding to a sensor. If the sensor is activated at least once during this time segment, the corresponding value is set to be 1; otherwise, is set to be 0. The label for each resident is set to be the label with the longest duration in this segment for each of them.

Activity classes and prototypes. In these two datasets, there are 27 labeled activity classes. Following the approaches adopted in [13, 149], we group similar activities into more general activities, and form 10 activities. In this way to let the activities we dealt

with to be more representative. When deciding these 10 activity classes, we refer to the way of grouping activities used in [13] and [149]. In addition, we refer to the taxonomy of activities listed in [38]. The 10 activities are: other, going out, preparing a meal (including preparing breakfast, preparing lunch, and preparing dinner), eating (including having breakfast, having lunch, having dinner, and having snack), doing cleaning (including washing dishes, laundry, and cleaning), sleeping/napping, relaxing (including watching TV, using Internet, reading book, and listening to music), studying, personal hygiene (including having shower, toileting, shaving, brushing teeth, and changing clothes), and social (including talking on the phone, having conversation, and having guest).

For the semantic space, we use the label-embedding space. As introduced in Section 2.2.2, this semantic space is based on the widely adopted word embedding technique in the area of natural language processing. In the label-embedding space, each word or phrase has a corresponding vector. The vector contains semantic information. It is an efficient way to get the semantic information about the seen and the unseen classes. In many works on zero-shot learning [52, 162, 165], label-embedding space is used as the semantic space. In our experiment, the label-embedding space we used is got by running the skip-gram model [103] on the English Wikipedia dump corpus² on December 1, 2017. We use Gensim [130] to perform this. The label-embedding space we got is of dimension 100. For activity labels with more than one word, we use the average of the word embedding vectors for each of the constituent word.

5.3.2.2 Evaluation Metrics

In smart home activity recognition problems, the distribution of classes is usually imbalanced [12, 20, 155]. During activity recognition, we are not only interested in activities with large number of instances, but also activities with small number of instances. In

²<https://dumps.wikimedia.org/>

Table 5.1: Number of days each activity occurs.

Activity	A1	A2	B1	B2	Total
other	30	24	30	25	109
going out	27	30	25	28	110
preparing a meal	30	14	23	1	68
eating	30	26	25	20	101
doing cleaning	30	21	13	0	64
sleeping/napping	30	27	28	25	110
relaxing	30	27	27	26	110
studying	18	2	19	2	41
personal hygiene	30	27	30	27	114
social	30	24	11	10	75

view of this, we adopt the metrics averaged over different classes:

$$Precision = \frac{1}{N} \sum_{i=1}^N \frac{TP_i}{TP_i + FP_i},$$

$$Recall = \frac{1}{N} \sum_{i=1}^N \frac{TP_i}{TP_i + FN_i},$$

and their harmonic mean

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall},$$

as the evaluation metrics, where N is the number of classes during training or testing, TP_i , FP_i , and FN_i are the number of true positive, false positive, and false negative instances for class i respectively. These metrics are not influenced by the distribution of classes.

5.3.3 Performance Evaluation

5.3.3.1 Evaluation Design

In each dataset, there are instances from 30 days. We use the instances from Day1 to Day24 as the training instances, and the instances from Day25 to Day30 as the testing instances. Then we decide the seen and the unseen activity classes. We perform this

based on the occurrence frequencies of the activities. Table 5.1 is the statistics of the number of days in which each activity occurs. In this table and in the following, we use A1 stands for Resident1 in HouseA, A2 stands for Resident2 in HouseA, B1 stands for Resident1 in HouseB, and B2 stands for Resident2 in HouseB. From this table we can see, the total number of occurrence days of the activities “other”, “going out”, “eating”, “sleeping/napping”, “relaxing”, and “personal hygiene” is much larger than that of the other four activities (preparing a meal, doing cleaning, studying, and social). This is in accordance with our common sense. These six activities are the activities that we almost need to perform every day, and they are more likely to be the activities of interest during training data labeling. We select these six as the seen activities. For training instances belonging to the other four activities, we relabel them as belonging to the “other” activity. The labels of the testing instances retain unchanged.

We do this evaluation design with the aim of to be in accordance with the practical applications. In practical applications, we often collect and label sensor readings of some days, and learn a classifier with them. In the following days, we use this classifier to classify the observed sensor readings. During data labeling in the training phase, just the predefined activities that are of interest are labeled, and all other readings are labeled as “other”. When we use this classifier, some of the activities we need to recognize may not be within the predefined activities and we also need to recognize them.

With this evaluation design, for each dataset, we have 34,560 training instances and 8,640 testing instances. In the training instances, 6 activity classes are labeled; in the testing instances, 10 activity classes are needed to classify. Each training instance contains the extracted features and the activity labels of the two residents.

5.3.3.2 Implementation Details

The nonlinear functions $\phi(\cdot)$, $\xi(\cdot)$ and $\theta(\cdot)$ in Equations 5.2 and 5.3 are implemented by the hyperbolic tangent function. Each prototype in the semantic space is ℓ_2 normalized.

Adam [84] is adopted as the optimization method. During model learning, we set the batch size to be 200, and the number of training epochs to be 300. The initial learning rate η is set to be 0.001. The dimension K of the output of the shared feature projection and the dimension H of the hidden space are determined through five-fold cross validation. The method is implemented using Theano [147].

5.3.3.3 Comparison Methods

Existing works on multi-resident activity recognition (reviewed in Section 2.1.4) are not under the setting in which there are unseen classes. In addition, existing works on multi-label zero-shot learning (reviewed in Section 2.2.3.3) are not for sensor-based activity recognition, and existing works on sensor-based zero-shot activity recognition (reviewed in Section 2.2.4) are for one-person setting. Thus, we do not have explicit existing methods to compare with. However, there exist some general approaches in multi-label classification [177], and they have been implemented in existing works on multi-resident activity recognition or multi-label zero-shot learning. We implement these approaches under our problem setting, and get the following methods. We compare our method with them. To let the comparison be more persuasive, we use the implementations similar to our method.

RMER: In this method, the recognition models are learned for each resident separately. When learning the recognition model for one resident, information about another resident in the training instances is regarded as noise [12]. This kind of method is widely used in multi-label classification problems [177], and is referred to as the overlaid method in [12]. Under our problem setting, we implement the RMER method using the same implementation of our method, with the difference that when learn the model for each resident, just the related one task is involved. We refer to this method as *respective model for each resident method (RMER)*.

CPAvg: In this method, the Cartesian product of the labels of the two residents are used as the new combined class labels. It converts the multi-label classification problem into a single-label classification problem. This approach is also widely used in multi-label classification problems. For the prototype of each combined class, we use the average of the prototypes of the two corresponding activity classes. We implement the CPAvg method using the same implementation of our method, with the difference that just one task is involved for the classification of the combined classes. We refer to this method as *Cartesian product and average prototype method (CPAvg)*.

CPCon: This method is similar to the CPAvg method. In this method, also the Cartesian product of the labels of the two residents are used as the new combined class labels. The difference of this method from the method CPAvg is that, in this method, for each combined class, the prototype is the concatenation of the prototypes of the two corresponding activity classes, instead of the average. We refer to this method as *Cartesian product and concatenation prototype method (CPCon)*.

5.3.3.4 Results and Analysis

The first four parts of Table 5.2 are the results of these methods. In this table, “avg A1 & A2” stands for the average of the results of A1 and A2, “avg B1 & B2” stands for the average of the results of B1 and B2. The best evaluation results among the four methods are shown in bold. From the results we can see, in general, compared with other three methods, our MRUA method achieves the best results. Even though a few evaluation results of our MRUA method is not the best, they are very close to the best results. Other methods cannot achieve this performance on all of the evaluation results. The results of all these four methods are significantly better than the random classification results of 10%. This illustrates the adopted zero-shot learning technique is effective under this problem setting, and the label-embedding space is an effective semantic space.

Table 5.2: Comparison of different methods (in %).

Method	House and Resident	Precision	Recall	F1-score
RMER	A1	39.57	34.91	37.09
	A2	34.83	32.47	33.61
	avg A1 & A2	37.20	33.69	35.35
	B1	46.29	45.64	45.96
	B2	57.58	48.07	52.40
	avg B1 & B2	51.94	46.85	49.18
CPAvg	A1	30.44	22.11	25.61
	A2	27.07	13.16	17.71
	avg A1 & A2	28.76	17.63	21.66
	B1	36.18	33.20	34.62
	B2	54.30	48.92	51.47
	avg B1 & B2	45.24	41.06	43.05
CPCon	A1	32.08	30.49	31.27
	A2	26.10	22.56	24.20
	avg A1 & A2	29.09	26.52	27.73
	B1	44.54	44.41	44.47
	B2	66.90	54.71	60.19
	avg B1 & B2	55.72	49.56	52.33
MRUA	A1	39.36	36.57	37.91
	A2	44.55	31.15	36.67
	avg A1 & A2	41.96	33.86	37.29
	B1	45.50	44.62	45.06
	B2	67.54	56.80	61.70
	avg B1 & B2	56.52	50.71	53.38
MRUA with all class labels	A1	52.49	48.20	50.25
	A2	37.91	34.31	36.02
	avg A1 & A2	45.20	41.25	43.14
	B1	61.96	63.88	62.91
	B2	69.40	64.27	66.73
	avg B1 & B2	65.68	64.08	64.82

In HouseA, the recognition results of A1 are better than that of A2 in all of these four methods. We consider this is because, in HouseA, A1’s duration of in the house is much longer than that of A2. In the training dataset, for A1, the number of instances belonging to the activity “going out” (not in the house) accounts 12.06% of the whole instances (there are 4,167 instances of A1 belonging to the activity “going out”, and the total number of training instances is 34,560). While for A2, this proportion is 48.47%

(there are 16,750 instances of A2 belonging to the activity “going out”, and the total number of training instances is 34,560). Thus, in HouseA, more sensor readings are about A1 and effective for activity recognition of A1. In HouseB, in all of these four methods, the recognition results of B2 are better than that of B1. In the training dataset, the going out time of B1 accounts 37.95% of the whole instances (there are 13,114 instances of B1 belonging to the activity “going out”, and the total number of training instances is 34,560). The going out time of B2 accounts 55.28% of the whole instances (there are 19,106 instances of B2 belonging to the activity “going out”, and the total number of training instances is 34,560). Although the going out time of B2 is longer than that of B1, the number of activities contained in the testing instances of B2 is only seven, with six among which are the seen classes. We consider this is the reason that the results of B2 are better than that of B1.

Among these four methods, the results of the CPAvg method are the lowest. We consider this is because, in this method, when forming the prototypes of the combined classes, the average of the prototypes of the two corresponding activity classes is used. In this way, information contained in these two prototypes has lost to some extent. This results in the poor performance. On the contrast, in the CPCon method, the prototypes of the combined classes are formed by concatenation, and the information loss is less. Thus, the results are much better. The results of MRUA and RMER for A1 and B1 are comparable. However, for A2 and B2, the results of MRUA are generally better than that of RMER. This shows that, by utilizing the multi-task learning strategy, our MRUA method is effective in improving the recognition results of residents with relatively less effective sensor readings.

To further analyze our method, we also compare it with the case in which in the training datasets, all of the ten activity classes are labeled. The results are shown at the bottom of Table 5.2. From the results we can see, with more labeled activity classes,

all of the recognition results improve a lot. Except the precision of A2. We consider this decrease is mainly because the duration of in the house for A2 is much shorter than that of A1. Thus, even more label information does not necessarily bring increase of the recognition results. The results show that, compared with the cases in which all of the classes are labeled during training, there exists room for improvement for the cases with the unseen classes.

5.3.4 Results on Each Testing Day

The testing results on each testing day are shown in Figure 5.2. From the figure we can see, for each house and each resident, the recognition results on different days vary. These results are in accordance with the results in [12]. It shows that, the activities of the residents are complex. In different days, the number of performed activities is different, and the way of performing the activities can also be different. This results in the differences on the recognition results on different days. For A2, the recognition results on Day29 and Day30 are extremely low. We consider this is because, in these two days, there is only one activity “going out” for this resident. The misclassification of some instances could cause the low performance.

5.3.5 Effect of Amount of Training Data

We evaluate the effect of the amount of training data by varying the number of days from which to get the training instances. When we use the training instances of n days, we randomly select n days from Day1 to Day24 to get the training instances. We repeat this ten times and report the average results.

The results are shown in Figure 5.3. From this figure we can see, the results of A1 and B1 are relatively stable. When the amount of training data increase to nine days for A1 and six days for B1, the results do not change much with the increase of the training data.

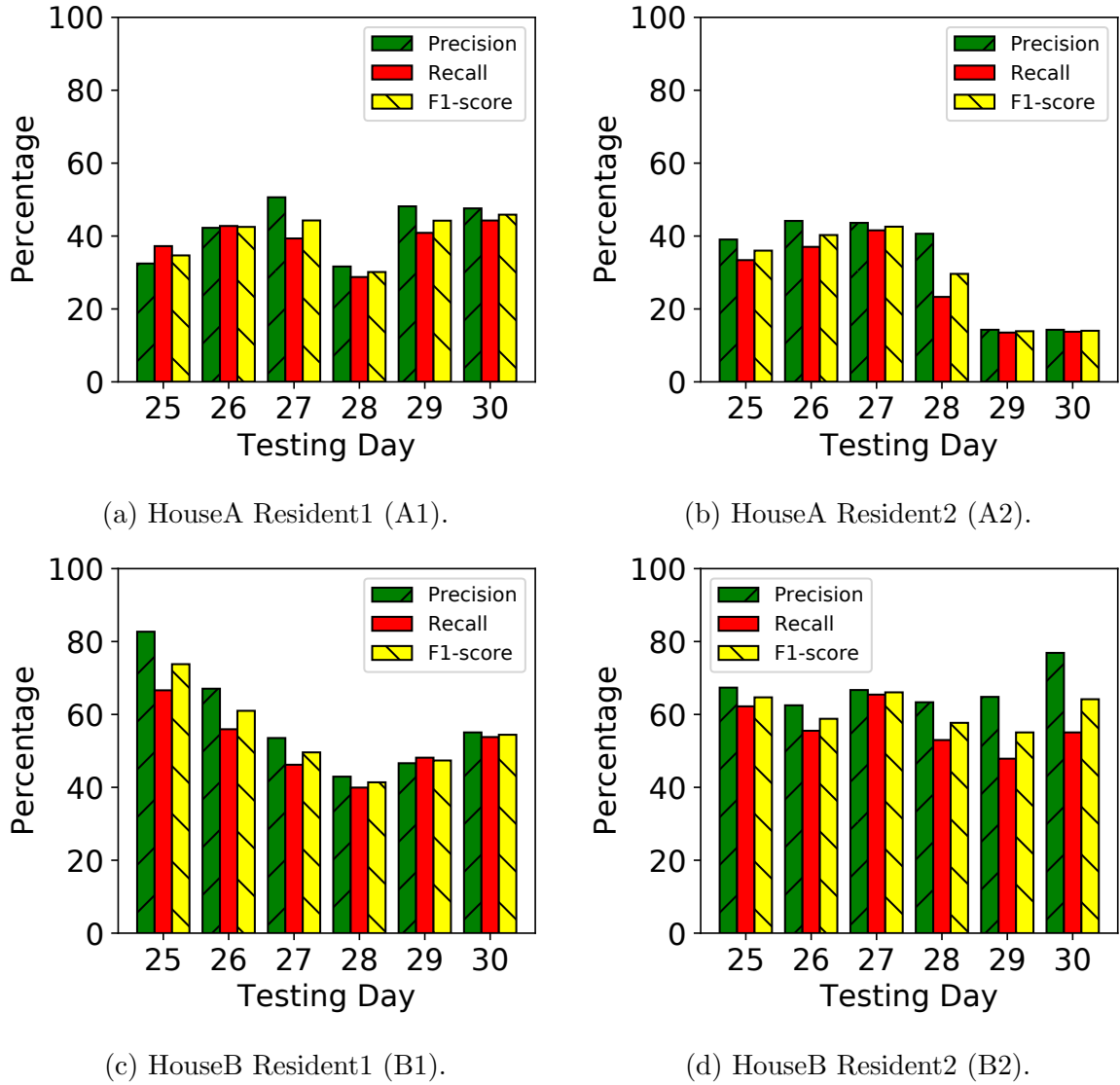


Figure 5.2: Results on each testing day.

These results are in accordance with the results in other smart home activity recognition problems [155]. However, for A2 and B2, the results are not so stable. We consider this difference is because, as stated in Section 5.3.3.4, in these two datasets, the duration of in the house of A2 and B2 is shorter than that of A1 and B1. Thus, with the increase of the training data, the effective amount of training instances for A2 and B2 are not necessarily increased as that of A1 and B1.

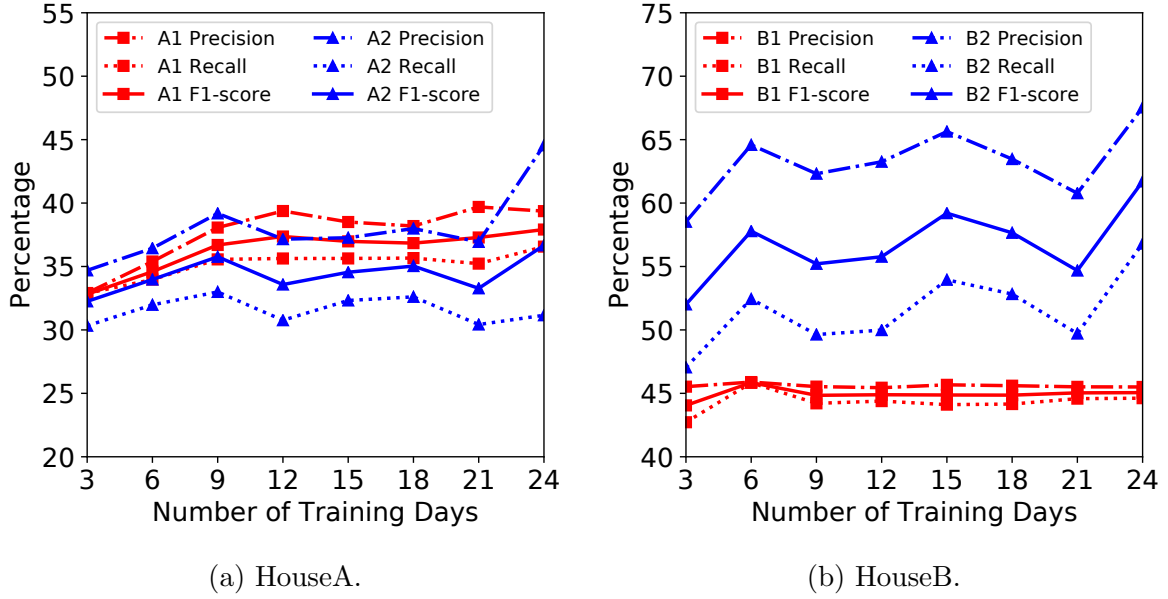


Figure 5.3: Effect of amount of training data.

5.3.6 Discussion

In the experiments, we show the effectiveness of our method. In our method, to implement zero-shot learning, the label-embedding space is utilized as the semantic space, and the compatibility function is used for classification. The hard/explicit parameter sharing structure is utilized to implement multi-task learning. From the results and the comparison with other methods, we can see the effectiveness of our implementation.

From the results, we can see there are still some aspects in which we can improve. Firstly, in a smart home environment, the recognition results for residents with relatively less sensor readings are generally not as good as that of the residents with more sensor readings. Approaches that can improve the recognition results for residents with less sensor readings can be explored. In addition, in the experiments, we can see when the training instances increase to certain amount, with the increase of the amount of the training instances, the recognition results do not prominently improve. Approaches that can make more use of the training instances can be explored.

In our problem, we focus on the problem setting in which, in the training data, at each time segment, we have the activity labels of all the residents. This is a common setting in many practical applications. Apart from it, in some applications, we may do not have the activity labels of all residents at the same time. In some time segments, we may just have the activity labels for one resident, and even in some time segments, we do not have activity labels for all of the residents. Under this problem setting, to represent the sensor readings, other data representation approaches may be needed (e.g., approaches that represent the data for each resident separately). Also recognition methods specific for this setting may be needed. In the future, we can explore approaches to address problems under this problem setting. Also, in our problem, when performing activity recognition, the number of residents is determined. In some applications, when performing activity recognition, the number of residents is not determined, needs to be detected by the method. In addition, in our problem, the deployed sensors in the smart home environment are fixed. In some applications, the deployed sensors can change dynamically. Approaches for problems under these situations can be explored as future works.

As another direction of future work, the deep learning approaches can be explored. Deep learning has achieved success in many areas. In deep learning based approaches, to learn the model, the amount of required training instances is large. However, under our problem setting, it is relatively difficult of get large amount of labeled training instances. This makes the deep learning based approaches not quite suitable. In the future, we will explore how to develop approaches that do not need large amount of training instances.

5.4 Summary

In this chapter, we study the problem of multi-resident activity recognition including unseen classes in smart homes. We propose a method MRUA to address this problem.

In our method, we regard the activity recognition of each resident as a learning task. With the multi-task learning structure, all tasks are learned jointly. Zero-shot learning techniques are used to let the model have the ability to recognize the unseen classes. To the best of our knowledge, this is the first work for multi-resident activity recognition with unseen classes in the smart homes. In the experiments, we evaluate the performance of the method, the results on each testing day, and the effect of amount of training data. The experiment results show that our method is effective.

Chapter 6

Conclusions and Future Work

In this chapter, we firstly conclude this thesis by summarizing the previous chapters. Then, we discuss some promising directions for future works.

6.1 Conclusions

In this thesis, we focus on the problems of sensor-based activity recognition under the setting of ARKN. Particularly, we focus on three problems: (1) unseen activity recognition for one person; (2) activity recognition across smart homes with different binary sensors and label spaces; and (3) multi-resident activity recognition including unseen classes in smart homes.

We give an introduction of the problem background in Chapter 1. And we also give a brief introduction of the three problems and the proposed methods. Then, in Chapter 2, we review related works in two areas: sensor-based activity recognition and zero-shot learning. In the following three chapters, we introduce our research works on the three problems.

In Chapter 3, we study the problem of unseen activity recognition for one person. It is a zero-shot learning problem. We propose a method NCBM to address this problem. Our method belongs to the compatibility based methods. In our method, a compatibility function is learned. With this compatibility function and the prototypes of the unseen

classes, nonlinear one-vs-rest classifiers for the unseen classes can be obtained. Our method is under the CIII learning setting, in which during model learning, no information about the testing instances and the unseen classes is needed. We conduct extensive experiments, and the experimental results show the effectiveness of our method.

In Chapter 4, we study the problem of activity recognition across smart homes with different binary sensors and label spaces. We propose a method BSST for this problem. In our method, based on the semantic information of the binary sensors, the characteristics of the binary sensors, and the time information, we extract instances from different sensor networks into a common feature space. Based on the semantic information of the activities, we define the prototypes of the class labels in the semantic space. Then, a compatibility function is learned for activity recognition. In our method, during model learning, no information about the target house is needed. Through experiments, we show the effectiveness of our method.

In Chapter 5, we study the problem of multi-resident activity recognition including unseen classes in smart homes. We propose a method MRUA to address this problem. In our method, the activity recognition of each resident is regarded as a learning task. And all tasks are learned jointly with multi-task learning techniques. In addition, zero-shot learning techniques are utilized. Thus, the unseen classes in the target label space can be recognized. The experiments show the effectiveness of our method.

6.2 Future Directions

For the problems of sensor-based activity recognition under the ARKN setting, we discuss some future directions.

- (i) **Sequential information.** The sensor readings are time series data. Thus, sequential information is contained within them. In this thesis, when we deal with

the sensor readings, we adopt the widely used sliding window strategy to separate the time series into segments. The sequential information is not effectively utilized. In problems under the supervised learning setting, sequential classification models like HMM [155], CRF [89], and RNN [70] have been widely used. And the sequential information contained in the sensor readings is utilized. On the other hand, there are some explorations [33] on the sequential model for sensor-based zero-shot activity recognition problems. In [33], CRF is used. Under the ARKN setting, as the classes needed to recognize including the unseen classes, it is more challenging than the supervised learning setting when utilizing the sequential information. As a future work, methods that can efficiently utilize the sequential information can be explored.

- (ii) **Data sources in the smart homes.** As introduced in Section 2.1.1.1, in smart homes, the ambient sensors are generally adopted as data collecting devices. Thus, in this thesis, we focus on smart homes deployed with ambient sensors. Particularly, in the work in Chapter 4, we focus on the binary sensors. In some applications in smart homes, there are different problem settings. For example, in some applications, both the ambient sensors and the wearable sensors are used; while in some applications, just the wearable sensors are used. These are different problem settings that can be explored in future works. In addition, in Chapter 4, when we consider the problem of activity recognition across different smart homes, just one source house is considered. As the future works, more than one source house can be considered. On the other hand, the problem setting which not just focusing on the binary sensors can be considered.
- (iii) **Semantic information.** The semantic spaces utilized in this thesis are the binary attribute spaces and the label-embedding spaces. When involving the attribute

spaces, as the problem setting is activity recognition, we use the common sense to define the attributes and the prototypes for the activities. As a future work, more systematic approaches to define the attributes can be explored. On the other hand, as introduced in Section 2.2, in zero-shot learning, there are various kinds of semantic spaces can be utilized. As a future work, we can explore the effect of other semantic spaces in the problems under ARKN setting.

- (iv) **Complex situations.** In some applications, the situation of activity recognition can be complex. For example, in some applications, there are concurrent activities [62], i.e., at a time point, there is more than one activity performed by a person. In some applications, there are interleaved activities [104], i.e., more than one activity is performed by a person with switching from one to another. In some applications, the hierarchy of the activities is considered [98]. These are more complex scenarios in activity recognition. As a future work, under the ARKN setting, applications in these scenarios can be considered, and learning methods designed specific for these scenarios can be explored.
- (v) **Combination with other learning problems.** In this thesis, we focus on the problem of activity recognition, which aims to recognize activity class labels from the sensor readings. In the research related with activity learning from sensor data [38], there also exist other problems. In the problem of activity discovery, the aim is to discover potential-activities/activity-patterns from the sensor readings. In the problem of activity prediction, the aim is not to recognize the activities, but to predict what activities the human will perform in the future. As future works, we can consider the combination of the ARKN setting problems with the activity discovery or activity prediction problems. This will have broader application scopes.

Appendix A

Class Splitting, Defined Attributes and Meta-Features

A.1 Splitting of Classes, Defined Attributes and Prototypes in Chapter 3

Table A.1 is the splitting of classes in dataset PAMAP2. Table A.2 is the splitting of classes in dataset OPP. Table A.3 is the splitting of classes in dataset TUD. Table A.4 is the defined attributes and prototypes for dataset PAMAP2. Table A.5 is the defined attributes and prototypes for dataset OPP.

A.2 Defined Meta-Features, Meta-Feature Vectors, Attributes, and Prototypes in Chapter 4

For the two datasets OrdonezA and OrdonezB, the defined meta-features and meta-feature vectors of the sensors are shown in Table A.6. In this table, for each sensor, in the brackets, “OA & OB” denotes this sensor is in both OrdonezA and OrdonezB, “OA” denotes this sensor is just in OrdonezA, and “OB” denotes this sensor is just in OrdonezB. For these two datasets, the defined attributes and prototypes of the activities are shown in Table A.7.

Table A.1: Class splitting in dataset PAMAP2.

Fold	Classes
fold 1	watching TV, house cleaning, standing, ascending stairs
fold 2	walking, rope jumping, sitting, descending stairs
fold 3	playing soccer, lying, vacuum cleaning, computer work
fold 4	cycling, running, Nordic walking
fold 5	ironing, car driving, folding laundry

Table A.2: Class splitting in dataset OPP.

Fold	Classes
fold 1	close drawer 2, clean table, toggle switch, open drawer 3
fold 2	open fridge, open door 2, close drawer 1, open drawer 2
fold 3	drink from cup, open drawer 1, close dishwasher
fold 4	close drawer 3, close door 2, open door 1
fold 5	close fridge, open dishwasher, close door 1

Table A.3: Class splitting in dataset TUD.

Fold	Classes
fold 1	making coffee, standing / talking, attending a presentation, discussing at whiteboard, fanning barbecue, running, walking while carrying something
fold 2	standing / talking on phone, standing / having a coffee, walking, wiping the whiteboard, queuing in line, sitting / talking on phone, brushing teeth
fold 3	driving bike, having breakfast, kneeling / doing something else, driving car, watching a movie, sitting / desk activities, standing / using the toilet
fold 4	washing hands, picking up cafeteria food, having lunch, personal hygiene, using the toilet, washing dishes
fold 5	walking freely, kneeling / making fire for barbecue, setting the table, lying while reading / using computer, sitting / having a coffee, having dinner

Table A.4: Defined attributes and prototypes for dataset PAMAP2.

Attribute aspect		Movement of body																				
Activity	Attribute	motion	static	cyclic motion	intense motion	translation motion	free motion	body vertical	body incline	body horizontal	body forward	body backward	body up	body down	body in place	torso transform	arms motion	arms static	arms bent	arms straight	arms bent-straight transform	hands hold something
lying		0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	1	1	1	0
sitting		0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	1	1	1	1	0
standing		0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	1	1	1	1	0
walking		1	0	1	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0
running		1	0	1	1	1	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	0
cycling		1	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	1	1	1	0	1
Nordic walking		1	0	1	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1	1
watching TV		0	1	0	0	0	0	1	1	0	0	0	0	0	1	0	1	1	1	1	1	0
computer work		0	1	0	0	0	0	1	1	0	0	0	0	0	1	0	1	1	1	1	1	1
car driving		0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	1
ascending stairs		1	0	1	0	0	0	1	1	0	1	0	1	0	0	0	1	0	1	1	1	1
descending stairs		1	0	1	0	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	1	1
vacuum cleaning		1	0	0	0	0	1	1	1	0	1	1	0	0	0	1	1	0	1	1	1	1
ironing		1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	1	1	1	1
folding laundry		1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	1	1	1	1
house cleaning		1	0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1
playing soccer		1	0	0	1	0	1	1	1	0	1	0	1	1	0	1	1	0	1	1	1	0
rope jumping		1	0	1	1	0	0	1	0	0	0	0	1	1	1	0	1	0	1	1	1	1

Table A.4: Defined attributes and prototypes for dataset PAMAP2 (continued).

Attribute aspect		Movement of body							Related object and environment													
Activity	Attribute	legs motion	legs static	legs bent	legs straight	legs bent-straight transform	legs alternate move forward	legs move up and/or down	seat	bike	poles	television	computer	car	stairs	vacuum	iron	clothes	soccer	rope	indoor	outdoor
lying		1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
sitting		0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1
standing		0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
walking		1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
running		1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
cycling		1	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1
Nordic walking		1	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
watching TV		1	1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0
computer work		1	1	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0
car driving		1	1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1
ascending stairs		1	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1
descending stairs		1	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1
vacuum cleaning		1	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
ironing		1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0
folding laundry		1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0
house cleaning		1	0	1	1	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	1	0
playing soccer		1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
rope jumping		1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1

Table A.5: Defined attributes and prototypes for dataset OPP.

Attribute aspect	Movement of body					Related object and environment									
Attribute Activity	open	close	clean	drink	toggle	door 1	door 2	fridge	dishwasher	drawer 1	drawer 2	drawer 3	table	cup	switch
open door 1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
open door 2	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
close door 1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
close door 2	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
open fridge	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
close fridge	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
open dishwasher	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
close dishwasher	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
open drawer 1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
close drawer 1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
open drawer 2	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
close drawer 2	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
open drawer 3	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0
close drawer 3	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
clean table	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
drink from cup	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0
toggle switch	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1

Table A.6: Defined meta-features and meta-feature vectors for datasets OrdonezA and OrdonezB.

Meta-feature aspect	Location					Function			
Meta-feature Sensor	living room	kitchen	bedroom	bathroom	entrance	meal related	bath related	toileting related	rest related
basin in bathroom (OA & OB)	0	0	0	1	0	0	0	0	0
bed in bedroom (OA & OB)	0	0	1	0	0	0	0	0	1
cabinet in bathroom (OA)	0	0	0	1	0	0	0	0	0
cooktop in kitchen (OA)	0	1	0	0	0	1	0	0	0
cupboard in kitchen (OA & OB)	0	1	0	0	0	1	0	0	0
fridge in kitchen (OA & OB)	0	1	0	0	0	1	0	0	0
maindoor in entrance (OA & OB)	0	0	0	0	1	0	0	0	0
microwave in kitchen (OA & OB)	0	1	0	0	0	1	0	0	0
seat in living room (OA & OB)	1	0	0	0	0	0	0	0	1
shower in bathroom (OA & OB)	0	0	0	1	0	0	1	0	0
toaster in kitchen (OA)	0	1	0	0	0	1	0	0	0
toilet in bathroom (OA & OB)	0	0	0	1	0	0	0	1	0
door in bedroom (OB)	0	0	1	0	0	0	0	0	1
door in kitchen (OB)	0	1	0	0	0	1	0	0	0
door in living room (OB)	1	0	0	0	0	0	0	0	0

Table A.7: Defined attributes and prototypes for datasets OrdonezA and OrdonezB.

Attribute aspect	Property				Time				Location								
Attribute Activity	meal related	bath related	toileting related	rest related	other	morning	noon	afternoon	evening	night	no specific time	living room	kitchen	bedroom	bathroom	outside	other
idle/other	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1
leave house	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0
toileting	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
taking a shower	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
sleeping	0	0	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0
preparing breakfast	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
preparing dinner	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
preparing lunch	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0
preparing snack	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
spare time/TV	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1
grooming	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1

Appendix B

List of Publications

- Wei Wang, Vincent W. Zheng, Han Yu, and Chunyan Miao. A Survey of Zero-Shot Learning: Settings, Methods, and Applications. *ACM Transactions on Intelligent Systems and Technology (ACM TIST)*, 10(2):13:1-13:37, 2019.
- Wei Wang and Chunyan Miao. Activity Recognition in New Smart Home Environments. *ACM International Conference on Multimedia Workshop: International Workshop on Multimedia for Personal Health and Health Care (HealthMedia 2018)*, ACM, pages 29-37, Seoul, Republic of Korea, October 22, 2018.
- Wei Wang and Chunyan Miao. Multi-Resident Activity Recognition with Unseen Classes in Smart Homes. *IEEE International Conference on Ubiquitous Intelligence and Computing (UIC 2018)*, IEEE, pages 780-787, Guangzhou, China, October 8-12, 2018.
- Wei Wang, Chunyan Miao, and Shuji Hao. Zero-Shot Human Activity Recognition via Nonlinear Compatibility Based Method. *International Conference on Web Intelligence (WI 2017)*, ACM, pages 322-330, Leipzig, Germany, August 23-26, 2017.
- Wei Wang, Han Yu, and Chunyan Miao. Deep Model for Dropout Prediction in MOOCs. *International Conference on Crowd Science and Engineering (ICCSE 2017)*, ACM, pages 26-32, Beijing, China, July 6-9, 2017.

References

- [1] Zahraa S. Abdallah, Mohamed Medhat Gaber, Bala Srinivasan, and Shonali Krishnaswamy. Anynovel: detection of novel concepts in evolving data streams. *Evolving Systems*, 7(2):73–93, 2016.
- [2] Zahraa S. Abdallah, Mohamed Medhat Gaber, Bala Srinivasan, and Shonali Krishnaswamy. Activity recognition with evolving data streams: A review. *ACM Computing Surveys (CSUR)*, 51(4):71, 2018.
- [3] Bilal M’hamed Abidine, Lamya Fergani, Belkacem Fergani, and Mourad Oussalah. The joint use of sequence features combination and modified weighted SVM for improving daily activity recognition. *Pattern Analysis and Applications*, 21(1):119–138, 2018.
- [4] Giovanni Acampora, Diane J. Cook, Parisa Rashidi, and Athanasios V. Vasilakos. A survey on ambient intelligence in healthcare. *Proceedings of the IEEE*, 101(12):2470–2494, 2013.
- [5] Zeynep Akata, Mateusz Malinowski, Mario Fritz, and Bernt Schiele. Multi-cue zero-shot learning with strong supervision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 59–68, 2016.
- [6] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for attribute-based classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 819–826, 2013.
- [7] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1425–1438, 2016.

- [8] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2927–2936, 2015.
- [9] Mohammad Al-Naser, Hiroki Ohashi, Sheraz Ahmed, Katsuyuki Nakamura, Takayuki Akiyama, Takuto Sato, Phong Nguyen, and Andreas Dengel. Hierarchical model for zero-shot activity recognition using wearable sensors. In *International Conference on Agents and Artificial Intelligence*, pages 478–485. INSTICC, SciTePress, 2018.
- [10] Mohammad Arif Ul Alam and Nirmalya Roy. Unseen activity recognitions: A hierarchical active transfer learning approach. In *IEEE International Conference on Distributed Computing Systems*, pages 436–446. IEEE, 2017.
- [11] Muhammad Raisul Alam, Mamun Bin Ibne Reaz, and Mohd Alauddin Mohd Ali. A review of smart homes - past, present, and future. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1190–1203, 2012.
- [12] Hande Alemdar and Cem Ersoy. Multi-resident activity tracking and recognition in smart environments. *Journal of Ambient Intelligence and Humanized Computing*, 8(4):513–529, 2017.
- [13] Hande Alemdar, Halil Ertan, Ozlem Durmaz Incel, and Cem Ersoy. ARAS human activity datasets in multiple homes with multiple residents. In *International Conference on Pervasive Computing Technologies for Healthcare and Workshops*, pages 232–235, 2013.
- [14] Hande Alemdar, Tim L.M. van Kasteren, and Cem Ersoy. Using active learning to allow activity recognition on a large scale. In *International Joint Conference on Ambient Intelligence*, pages 105–114. Springer, 2011.
- [15] Kerem Altun, Billur Barshan, and Orkun Tunçel. Comparative study on classifying human activities with miniature inertial and magnetic sensors. *Pattern Recognition*, 43(10):3605–3620, 2010.

- [16] Akin Avci, Stephan Bosch, Mihai Marin-Perianu, Raluca Marin-Perianu, and Paul Havinga. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In *International Conference on Architecture of Computing Systems Workshop*, pages 167–176. VDE, 2010.
- [17] Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *IEEE International Conference on Computer Vision*, pages 4247–4255, 2015.
- [18] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. In *International Conference on Pervasive Computing*, pages 1–17. Springer, 2004.
- [19] Asma Benmansour, Abdelhamid Bouchachia, and Mohammed Feham. Multioccupant activity recognition in pervasive smart home environments. *ACM Computing Surveys (CSUR)*, 48(3):34, 2016.
- [20] Asma Benmansour, Abdelhamid Bouchachia, and Mohammed Feham. Modeling interaction in multi-resident activities. *Neurocomputing*, 230:133–142, 2017.
- [21] Ulf Blanke and Bernt Schiele. Remember and transfer what you have learned—recognizing composite activities based on activity spotting. In *International Symposium on Wearable Computers*, pages 1–8. IEEE, 2010.
- [22] Bruno Bouchard, Sylvain Giroux, and Abdenour Bouzouane. A smart home agent for plan recognition of cognitively-impaired patients. *Journal of Computers*, 1(5):53–62, 2006.
- [23] Maxime Bucher, Stéphane Herbin, and Frédéric Jurie. Generating visual representations for zero-shot classification. In *ICCV Workshops*, pages 2666–2673, 2017.
- [24] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):33:1–33:33, 2014.

- [25] Spencer Cappallo, Thomas Mensink, and Cees G. M. Snoek. Image2Emoji: Zero-shot emoji prediction for visual media. In *ACM International Conference on Multimedia*, pages 1311–1314, 2015.
- [26] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.
- [27] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5327–5336, 2016.
- [28] Wei-Lun Chao, Soravit Changpinyo, Boqing Gong, and Fei Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *European Conference on Computer Vision*, pages 52–68, 2016.
- [29] Liming Chen, Jesse Hoey, Chris D. Nugent, Diane J. Cook, and Zhiwen Yu. Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):790–808, 2012.
- [30] Rong Chen and Yu Tong. A two-stage method for solving multi-resident activity recognition in smart environments. *Entropy*, 16(4):2184–2203, 2014.
- [31] Wen-Hui Chen and Yiran Chen. An ensemble approach to activity recognition based on binary sensor readings. In *IEEE International Conference on e-Health Networking, Applications and Services*, pages 1–5, 2017.
- [32] Heng-Tze Cheng. *Learning and Recognizing the Hierarchical and Sequential Structure of Human Activities*. PhD thesis, 2013.
- [33] Heng-Tze Cheng, Martin Griss, Paul Davis, Jianguo Li, and Di You. Towards zero-shot learning for human activity recognition using semantic attribute sequence model. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 355–358. ACM, 2013.
- [34] Heng-Tze Cheng, Feng-Tso Sun, Martin Griss, Paul Davis, Jianguo Li, and Di You. NuActiv: Recognizing unseen new activities using semantic attribute-based learning. In *International Conference on Mobile Systems, Applications, and Services*, pages 361–374. ACM, 2013.

- [35] Yi-ting Chiang and Jane Yung-jen Hsu. Knowledge transfer in activity recognition using sensor profile. In *9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*, pages 180–187, 2012.
- [36] Yi-Ting Chiang, Kuo-Chung Hsu, Ching-Hu Lu, Li-Chen Fu, and Jane Yung-Jen Hsu. Interaction models for multiple-resident activity recognition in a smart home. In *International Conference on Intelligent Robots and Systems*, pages 3753–3758. IEEE, 2010.
- [37] Diane Cook, Kyle D. Feuz, and Narayanan C. Krishnan. Transfer learning for activity recognition: a survey. *Knowledge and Information Systems*, 36(3):537–556, 2013.
- [38] Diane J. Cook and Narayanan C. Krishnan. *Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data*. John Wiley & Sons, 2015.
- [39] Diane J. Cook, Narayanan C. Krishnan, and Parisa Rashidi. Activity discovery and activity recognition: A new partnership. *IEEE Transactions on Cybernetics*, 43(3):820–828, 2013.
- [40] Aaron S. Crandall and Diane J. Cook. Coping with multiple residents in a smart environment. *Journal of Ambient Intelligence and Smart Environments*, 1(4):323–334, 2009.
- [41] Aaron S. Crandall and Diane J. Cook. Using a hidden markov model for resident identification. In *International Conference on Intelligent Environments*, pages 74–79. IEEE, 2010.
- [42] Jeffrey Dalton, James Allan, and Pranav Mirajkar. Zero-shot video retrieval using content and concepts. In *ACM International Conference on Information and Knowledge Management*, pages 1857–1860, 2013.

- [43] Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. Large-scale object classification using label relation graphs. In *European Conference on Computer Vision*, pages 48–64, 2014.
- [44] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.
- [45] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.
- [46] Mohamed Elhoseiny, Babak Saleh, and Ahmed Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *IEEE International Conference on Computer Vision*, pages 2584–2591, 2013.
- [47] Mohamed Elhoseiny, Yizhe Zhu, Han Zhang, and Ahmed Elgammal. Link the head to the “beak”: Zero shot learning from noisy text description at part precision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6288–6297, 2017.
- [48] Ifat Afrin Emi and John A. Stankovic. SARRIMA: Smart ADL recognizer and resident identifier in multi-resident accommodations. In *Wireless Health Conference*, pages 4:1–4:8. ACM, 2015.
- [49] Macarena Espinilla, Javier Medina, Josef Hallberg, and Chris Nugent. A new approach based on temporal sub-windows for online sensor-based activity recognition. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–13, 2018.
- [50] Kyle D. Feuz and Diane J. Cook. Transfer learning across feature-rich heterogeneous feature spaces via feature-space remapping (FSR). *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(1):3:1–3:27, 2015.
- [51] Anthony Fleury, Michel Vacher, and Norbert Noury. SVM-based multimodal classification of activities of daily living in health smart homes: Sensors, algorithms, and first experimental results. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):274–283, 2010.

- [52] Andrea Frome, Greg S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. DeViSE: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, pages 2121–2129, 2013.
- [53] Yanwei Fu, Timothy M. Hospedales, Tao Xiang, and Shaogang Gong. Learning multimodal latent attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(2):303–316, 2014.
- [54] Yanwei Fu, Yongxin Yang, Timothy Hospedales, Tao Xiang, and Shaogang Gong. Transductive multi-label zero-shot learning. In *British Machine Vision Conference*, 2014.
- [55] Zhenyong Fu, Tao Xiang, Elyor Kodirov, and Shaogang Gong. Zero-shot object recognition by semantic manifold distance. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2644, 2015.
- [56] Chuang Gan, Ming Lin, Yi Yang, Yueting Zhuang, and Alexander G. Hauptmann. Exploring semantic inter-class relationships (SIR) for zero-shot action recognition. In *AAAI Conference on Artificial Intelligence*, pages 3769–3775, 2015.
- [57] Chuang Gan, Tianbao Yang, and Boqing Gong. Learning attributes equals multi-source domain generalization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 87–97, 2016.
- [58] Chuang Gan, Yi Yang, Linchao Zhu, Deli Zhao, and Yueting Zhuang. Recognizing an action using its name: A knowledge-based approach. *International Journal of Computer Vision*, 120(1):61–77, 2016.
- [59] Lianli Gao, Jingkuan Song, Junming Shao, Xiaofeng Zhu, and Heng Tao Shen. Zero-shot image categorization by image correlation exploration. In *International Conference on Multimedia Retrieval*, pages 487–490, 2015.
- [60] Salvador García, Julián Luengo, and Francisco Herrera. *Data Preprocessing in Data Mining*. Springer, 2015.

- [61] Pinghua Gong, Jieping Ye, and Changshui Zhang. Robust multi-task feature learning. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 895–903. ACM, 2012.
- [62] Tao Gu, Zhanqing Wu, Xianping Tao, Hung Keng Pung, and Jian Lu. epSICAR: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In *International Conference on Pervasive Computing and Communications*, pages 1–9. IEEE, 2009.
- [63] Tao Gu, Zhanqing Wu, Liang Wang, Xianping Tao, and Jian Lu. Mining emerging patterns for recognizing activities of multiple users in pervasive computing. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 1–10. IEEE, 2009.
- [64] Donghai Guan, Weiwei Yuan, Young-Koo Lee, Andrey Gavrilov, and Sungyoung Lee. Activity recognition based on semi-supervised learning. In *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 469–475. IEEE, 2007.
- [65] Yuchen Guo, Guiguang Ding, Jungong Han, and Yue Gao. Synthesizing samples for zero-shot learning. In *International Joint Conference on Artificial Intelligence*, pages 1774–1780, 2017.
- [66] Yuchen Guo, Guiguang Ding, Jungong Han, and Yue Gao. Zero-shot learning with transferred samples. *IEEE Transactions on Image Processing*, 26(7):3277–3290, 2017.
- [67] Yuchen Guo, Guiguang Ding, Jungong Han, and Yue Gao. Zero-shot recognition via direct classifier learning with transferred samples and pseudo labels. In *AAAI Conference on Artificial Intelligence*, pages 4061–4067, 2017.
- [68] Yuchen Guo, Guiguang Ding, Xiaoming Jin, and Jianmin Wang. Transductive zero-shot recognition via shared model space learning. In *AAAI Conference on Artificial Intelligence*, 2016.

- [69] Hirotaka Hachiya, Masashi Sugiyama, and Naonori Ueda. Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition. *Neurocomputing*, 80:93–101, 2012.
- [70] Nils Y. Hammerla, Shane Halloran, and Thomas Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *International Joint Conference on Artificial Intelligence*, pages 1533–1540, 2016.
- [71] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Elsevier, 2012.
- [72] H M Sajjad Hossain, Nirmalya Roy, and Md Abdullah Al Hafiz Khan. Active learning enabled activity recognition. In *IEEE International Conference on Pervasive Computing and Communications*, 2016.
- [73] Kuo-Chung Hsu, Yi-Ting Chiang, Gu-Yang Lin, Ching-Hu Lu, Jane Yung-Jen Hsu, and Li-Chen Fu. Strategies for inference mechanism of conditional random fields for multiple-resident activity recognition in a smart home. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 417–426. Springer, 2010.
- [74] Derek Hao Hu and Qiang Yang. Transfer learning for activity recognition via sensor mapping. In *International Joint Conference on Artificial Intelligence*, pages 1962–1967, 2011.
- [75] Tâm Huynh, Mario Fritz, and Bernt Schiele. Discovery of activity patterns using topic models. In *International Conference on Ubiquitous Computing*, pages 10–19. ACM, 2008.
- [76] Isibor Kennedy Ihianle, Usman Naeem, and Abdel-Rahman Tawil. Recognition of activities of daily living from topic model. *Procedia Computer Science*, 98:24–31, 2016.
- [77] Mihir Jain, Jan C. van Gemert, Thomas Mensink, and Cees G.M. Snoek. Objects2action: Classifying and localizing actions without any video example. In *IEEE International Conference on Computer Vision*, pages 4588–4596, 2015.

- [78] Dinesh Jayaraman and Kristen Grauman. Zero-shot recognition with unreliable attributes. In *Advances in Neural Information Processing Systems*, pages 3464–3472, 2014.
- [79] Huajie Jiang, Ruiping Wang, Shiguang Shan, Yi Yang, and Xilin Chen. Learning discriminative latent attributes for zero-shot classification. In *IEEE International Conference on Computer Vision*, pages 4233–4242, 2017.
- [80] Shian-Ru Ke, Hoang Le Uyen Thuc, Yong-Jin Lee, Jenq-Neng Hwang, Jang-Hee Yoo, and Kyoung-Ho Choi. A review on video-based human activity recognition. *Computers*, 2(2):88–131, 2013.
- [81] Md Abdullah Al Hafiz Khan and Nirmalya Roy. TransAct: Transfer learning enabled activity recognition. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 545–550. IEEE, 2017.
- [82] Md Abdullah Al Hafiz Khan and Nirmalya Roy. UnTran: Recognizing unseen activities with unlabeled data using transfer learning. In *IEEE/ACM International Conference on Internet-of-Things Design and Implementation*, pages 37–47. IEEE, 2018.
- [83] Eunju Kim, Sumi Helal, and Diane Cook. Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9(1):48–53, 2010.
- [84] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.
- [85] Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. Unsupervised domain adaptation for zero-shot learning. In *IEEE International Conference on Computer Vision*, pages 2452–2460, 2015.
- [86] Elyor Kodirov, Tao Xiang, and Shaogang Gong. Semantic autoencoder for zero-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4447–4456, 2017.

- [87] Svetlana Kordumova, Thomas Mensink, and Cees G. M. Snoek. Pooling objects for recognizing scenes without examples. In *International Conference on Multimedia Retrieval*, pages 143–150, 2016.
- [88] S.B. Kotsiantis, D. Kanellopoulos, and P.E. Pintelas. Data preprocessing for supervised learning. *International Journal of Computer Science*, 1(2):111–117, 2006.
- [89] Narayanan C Krishnan and Diane J Cook. Activity recognition on streaming sensor data. *Pervasive and Mobile Computing*, 10:138–154, 2014.
- [90] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958, 2009.
- [91] Oscar D. Lara and Miguel A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys and Tutorials*, 15(3):1192–1209, 2013.
- [92] Chung-Wei Lee, Wei Fang, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. Multi-label zero-shot learning with structured knowledge graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1576–1585, 2018.
- [93] Hanhui Li, Donghui Li, and Xiaonan Luo. BAP: Bimodal attribute prediction for zero-shot image categorization. In *ACM International Conference on Multimedia*, pages 1013–1016, 2014.
- [94] Xin Li, Yuhong Guo, and Dale Schuurmans. Semi-supervised zero-shot classification with label representation learning. In *IEEE International Conference on Computer Vision*, pages 4211–4219, 2015.
- [95] Yanan Li, Donghui Wang, Huanhang Hu, Yuetan Lin, and Yueting Zhuang. Zero-shot recognition using dual visual-semantic mapping paths. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5207–5215, 2017.
- [96] Kongming Liang, Hong Chang, Shiguang Shan, and Xilin Chen. A unified multiplicative framework for attribute learning. In *IEEE International Conference on Computer Vision*, pages 2506–2514, 2015.

- [97] Jingen Liu, Benjamin Kuipers, and Silvio Savarese. Recognizing human actions by attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3337–3344, 2011.
- [98] Ye Liu, Liqiang Nie, Lei Han, Luming Zhang, and David S. Rosenblum. Action2Activity: Recognizing complex activities from sensor data. In *International Joint Conference on Artificial Intelligence*, volume 2015, pages 1617–1623, 2015.
- [99] Yukun Ma, Erik Cambria, and Sa Gao. Label embedding for zero-shot fine-grained named entity typing. In *International Conference on Computational Linguistics*, pages 171–180, 2016.
- [100] Moe Matsuki and Sozo Inoue. Recognizing unknown activities using semantic word vectors and twitter timestamps. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 823–830. ACM, 2016.
- [101] Uwe Maurer, Asim Smailagic, Daniel P. Siewiorek, and Michael Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *International Workshop on Wearable and Implantable Body Sensor Networks*, pages 113–116. IEEE, 2006.
- [102] Thomas Mensink, Efstratios Gavves, and Cees G.M. Snoek. Costa: Co-occurrence statistics for zero-shot classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2441–2448, 2014.
- [103] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR Workshops*, 2013.
- [104] Joseph Modayil, Tongxin Bai, and Henry Kautz. Improving the recognition of interleaved activities. In *International Conference on Ubiquitous Computing*, pages 40–43. ACM, 2008.
- [105] Tanmoy Mukherjee and Timothy Hospedales. Gaussian visual-linguistic embedding for zero-shot recognition. In *2016 Conference on Empirical Methods in Natural Language Processing*, pages 912–918, 2016.

- [106] Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J. Kim, and Johannes Fürnkranz. Predicting unseen labels using label hierarchies in large-scale multi-label learning. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 102–118, 2015.
- [107] Le T. Nguyen, Patrick Tague, Ming Zeng, and Joy Zhang. SuperAD: Supervised activity discovery. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pages 1463–1472. ACM, 2015.
- [108] Le T. Nguyen, Ming Zeng, Patrick Tague, and Joy Zhang. I did not smoke 100 cigarettes today! avoiding false positives in real-world activity recognition. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1053–1063. ACM, 2015.
- [109] Le T. Nguyen, Ming Zeng, Patrick Tague, and Joy Zhang. Recognizing new activities with limited training data. In *ACM International Symposium on Wearable Computers*, pages 67–74. ACM, 2015.
- [110] Tuan Anh Nguyen, Andrea Raspitzu, and Marco Aiello. Ontology-based office activity recognition with applications for energy savings. *Journal of Ambient Intelligence and Humanized Computing*, 5(5):667–681, 2014.
- [111] Hiroki Ohashi, Mohammad Al-Naser, Sheraz Ahmed, Katsuyuki Nakamura, Takuto Sato, and Andreas Dengel. Attributes’ importance for zero-shot pose-classification based on wearable sensors. *Sensors*, 18(8):2485, 2018.
- [112] Fco Javier Ordóñez, Paula de Toledo, and Araceli Sanchis. Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. *Sensors*, 13(5):5460–5477, 2013.
- [113] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

- [114] Mark Palatucci, Dean Pomerleau, Geoffrey Hinton, and Tom M. Mitchell. Zero-shot learning with semantic output codes. In *Advances in Neural Information Processing Systems*, pages 1410–1418, 2009.
- [115] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- [116] Devi Parikh and Kristen Grauman. Relative attributes. In *IEEE International Conference on Computer Vision*, pages 503–510, 2011.
- [117] Liangying Peng, Ling Chen, Zhenan Ye, and Yi Zhang. AROMA: A deep multi-task learning based simple and complex human activity recognition method using wearable sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(2):74:1–74:16, 2018.
- [118] Mike Perkowitz, Matthai Philipose, Kenneth Fishkin, and Donald J. Patterson. Mining models of human activities from the web. In *International Conference on World Wide Web*, pages 573–582. ACM, 2004.
- [119] Te Pi, Xi Li, and Zhongfei (Mark) Zhang. Boosted zero-shot learning with semantic correlation regularization. In *International Joint Conference on Artificial Intelligence*, pages 2599–2605, 2017.
- [120] Susanna Pirttikangas, Kaori Fujinami, and Tatsuo Nakajima. Feature selection and activity recognition from wearable sensors. In *International Symposium on Ubiquitous Computing Systems*, pages 516–527. Springer, 2006.
- [121] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, 2010.
- [122] Markus Prosegger and Abdelhamid Bouchachia. Multi-resident activity recognition using incremental decision trees. In *International Conference on Adaptive and Intelligent Systems*, pages 182–191. Springer, 2014.

- [123] Ruizhi Qiao, Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. Less is more: Zero-shot learning from online textual documents with noise suppression. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2249–2257, 2016.
- [124] Jie Qin, Li Liu, Ling Shao, Fumin Shen, Bingbing Ni, Jiaxin Chen, and Yunhong Wang. Zero-shot action recognition with error-correcting output codes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1042–1051, 2017.
- [125] Vijay Rajanna, Raniero Lara-Garduno, Dev Jyoti Behera, Karthic Madanagopal, Daniel Goldberg, and Tracy Hammond. Step up life: A context aware health assistant. In *ACM SIGSPATIAL International Workshop on the Use of GIS in Public Health*, pages 21–30. ACM, 2014.
- [126] Parisa Rashidi and Diane J. Cook. Transferring learned activities in smart environments. In *International Conference on Intelligent Environments*, pages 185–192, 2009.
- [127] Parisa Rashidi and Diane J. Cook. Activity recognition based on home to home transfer learning. In *AAAI Workshop on Plan, Activity, and Intent Recognition*, 2010.
- [128] Parisa Rashidi, Diane J. Cook, Lawrence B. Holder, and Maureen Schmitter-Edgecombe. Discovering activities to recognize and track in a smart environment. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):527–539, 2011.
- [129] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–58, 2016.
- [130] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50. ELRA, 2010.

- [131] Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In *International Symposium on Wearable Computers*, pages 108–109, 2012.
- [132] Zhou Ren, Hailin Jin, Zhe Lin, Chen Fang, and Alan Yuille. Joint image-text representation by gaussian visual-semantic embedding. In *ACM International Conference on Multimedia*, pages 207–211, 2016.
- [133] Zhou Ren, Hailin Jin, Zhe Lin, Chen Fang, and Alan Yuille. Multiple instance visual-semantic embedding. In *British Machine Vision Conference*, 2017.
- [134] Daniele Riboni, Timo Sztyler, Gabriele Civitarese, and Heiner Stuckenschmidt. Unsupervised recognition of interleaved activities of daily living through ontological and probabilistic reasoning. In *International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1–12. ACM, 2016.
- [135] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczech, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkel, Alois Ferscha, Jakob Doppler, Clemens Holzmann, Marc Kurz, Gerald Holl, Ricardo Chavarriaga, Hesam Sagha, Hamidreza Bayati, Marco Creatura, and José del R. Millán. Collecting complex activity datasets in highly rich networked sensor environments. In *International Conference on Networked Sensing Systems*, pages 233–240. IEEE, 2010.
- [136] Marcus Rohrbach, Sandra Ebert, and Bernt Schiele. Transfer learning in a transductive setting. In *Advances in Neural Information Processing Systems*, pages 46–54, 2013.
- [137] Marcus Rohrbach, Michael Stark, and Bernt Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1641–1648, 2011.
- [138] Marcus Rohrbach, Michael Stark, György Szarvas, Iryna Gurevych, and Bernt Schiele. What helps where - and why? semantic relatedness for knowledge transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 910–917, 2010.

- [139] Bernardino Romera-Paredes and Philip H.S. Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161, 2015.
- [140] Nirmalya Roy, Archan Misra, and Diane Cook. Ambient and smartphone sensor assisted ADL recognition in multi-inhabitant smart environments. *Journal of Ambient Intelligence and Humanized Computing*, 7(1):1–19, 2016.
- [141] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.
- [142] Geetika Singla, Diane J Cook, and Maureen Schmitter-Edgecombe. Recognizing independent and joint activities among multiple residents in smart environments. *Journal of Ambient Intelligence and Humanized Computing*, 1(1):57–63, 2010.
- [143] Richard Socher, Milind Ganjoo, Christopher D. Manning, and Andrew Y. Ng. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems*, pages 935–943, 2013.
- [144] Maja Stikic, Kristof Van Laerhoven, and Bernt Schiele. Exploring semi-supervised and active learning for activity recognition. In *International Symposium on Wearable Computers*, pages 81–88, 2008.
- [145] Sanatan Sukhija, Narayanan C. Krishnan, and Gurkanwal Singh. Supervised heterogeneous domain adaptation via random forests. In *International Joint Conference on Artificial Intelligence*, pages 2039–2045, 2016.
- [146] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *International Conference on Pervasive Computing*, pages 158–175. Springer, 2004.
- [147] Theano Development Team. Theano: A python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, 2016.
- [148] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

- [149] Can Tunca, Hande Alemdar, Halil Ertan, Ozlem Durmaz Incel, and Cem Ersoy. Multimodal wireless sensor network-based ambient assisted living in real homes with multiple residents. *Sensors*, 14(6):9692–9719, 2014.
- [150] Pavan Turaga, Rama Chellappa, V. S. Subrahmanian, and Octavian Udrea. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1473–1488, 2008.
- [151] Naman Turakhia and Devi Parikh. Attribute dominance: What pops out? In *IEEE International Conference on Computer Vision*, pages 1225–1232, 2013.
- [152] Yonatan Vaizman, Nadir Weibel, and Gert Lanckriet. Context recognition in-the-wild: Unified model for multi-modal sensors and multi-label classification. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):168:1–168:22, 2017.
- [153] Tim van Kasteren, Gwenn Englebienne, and Ben J.A. Kröse. Recognizing activities in multiple contexts using transfer learning. In *AAAI Fall Symposium: AI in Eldercare: New Solutions to Old Problems*, pages 142–149, 2008.
- [154] Tim van Kasteren, Gwenn Englebienne, and Ben J.A. Kröse. Transferring knowledge of activity recognition across sensor networks. In *International Conference on Pervasive Computing*, pages 283–300, 2010.
- [155] Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. Accurate activity recognition in a home setting. In *International Conference on Ubiquitous Computing*, pages 1–9. ACM, 2008.
- [156] Vinay Kumar Verma, Gundeep Arora, Ashish Mishra, and Piyush Rai. Generalized zero-shot learning via synthesized examples. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4281–4289, 2018.
- [157] Vinay Kumar Verma and Piyush Rai. A simple exponential family framework for zero-shot learning. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 792–808, 2017.

- [158] Sarvesh Vishwakarma and Anupam Agrawal. A survey on activity recognition and behavior understanding in video surveillance. *The Visual Computer*, 29(10):983–1009, 2013.
- [159] Donghui Wang, Yanan Li, Yuetan Lin Lin, and Yueting Zhuang. Relational knowledge transfer for zero-shot learning. In *AAAI Conference on Artificial Intelligence*, pages 2145–2151, 2016.
- [160] Liang Wang, Tao Gu, Xianping Tao, and Jian Lu. Sensor-based human activity recognition in a multi-user scenario. In *European Conference on Ambient Intelligence*, pages 78–87. Springer, 2009.
- [161] Qian Wang and Ke Chen. Alternative semantic representations for zero-shot human action recognition. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 87–102, 2017.
- [162] Qian Wang and Ke Chen. Zero-shot visual recognition via bidirectional latent embedding. *International Journal of Computer Vision*, 124(3):356–383, 2017.
- [163] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6857–6866, 2018.
- [164] Danny Wyatt, Matthai Philipose, and Tanzeem Choudhury. Unsupervised activity recognition using automatically mined common sense. In *The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, pages 21–27, 2005.
- [165] Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 69–77, 2016.
- [166] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. *arXiv preprint arXiv:1707.00600v3*, 2017.

- [167] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5542–5551, 2018.
- [168] Xing Xu, Fumin Shen, Yang Yang, Jie Shao, and Zi Huang. Transductive visual-semantic embedding for zero-shot learning. In *International Conference on Multimedia Retrieval*, pages 41–49, 2017.
- [169] Xing Xu, Fumin Shen, Yang Yang, Dongxiang Zhang, Heng Tao Shen, and Jingkuan Song. Matrix tri-factorization with manifold regularizations for zero-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2007–2016, 2017.
- [170] Xun Xu, Timothy Hospedales, and Shaogang Gong. Transductive zero-shot action recognition by word-vector embedding. *International Journal of Computer Vision*, 123(3):309–333, 2017.
- [171] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *International Joint Conference on Artificial Intelligence*, pages 3995–4001, 2015.
- [172] Lina Yao, Feiping Nie, Quan Z. Sheng, Tao Gu, Xue Li, and Sen Wang. Learning from less for better: Semi-supervised activity recognition via shared structure discovery. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 13–24. ACM, 2016.
- [173] Majid Yazdani and James Henderson. A model of zero-shot learning of spoken language understanding. In *2015 Conference on Empirical Methods in Natural Language Processing*, pages 244–249, 2015.
- [174] Juan Ye, Lei Fang, and Simon Dobson. Discovery and recognition of unknown activities. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 783–792. ACM, 2016.

- [175] Meng Ye and Yuhong Guo. Zero-shot classification with discriminative semantic representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5103–5111, 2017.
- [176] Li Zhang, Tao Xiang, and Shaogang Gong. Learning a deep embedding model for zero-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3010–3019, 2017.
- [177] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.
- [178] Yang Zhang, Boqing Gong, and Mubarak Shah. Fast zero-shot image tagging. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5985–5994, 2016.
- [179] Bo Zhao, Botong Wu, Tianfu Wu, and Yizhou Wang. Zero-shot learning posed as a missing data problem. In *ICCV Workshops*, pages 2616–2622, 2017.
- [180] Vincent W. Zheng, Derek Hao Hu, and Qiang Yang. Cross-domain activity recognition. In *International Conference on Ubiquitous Computing*, pages 61–70, 2009.
- [181] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.