Building occupant sensing : occupancy prediction and behavior recognition

Zhu, Qingchang

2018

Zhu, Q. (2018). Building occupant sensing : occupancy prediction and behavior recognition.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/88236>

<https://doi.org/10.32657/10220/45814>

Building Occupant Sensing: Occupancy Prediction and Behavior Recognition

Qingchang Zhu

School of Electrical & Electronic Engineering

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

2018

Statement of Originality

I hereby certify that the intellectual content of this thesis is the product of my original research work and has not been submitted for a higher degree to any other University or Institution.

.....

Date

.....

Qingchang Zhu

Acknowledgements

Firstly, I would like to thank my supervisor, Professor Yeng Chai Soh, for his patience and advice on how to conduct quality research. He has taught me how to formulate research problems and to present the novelty of ideas in a clear and concise manner. In addition, I would also like to express my gratitude to Professor Lihua Xie for his advice during the weekly group seminars and discussions.

I appreciate the collaboration with Dr. Zhenghua Chen. He shared his perseverance and diligence with me and we have a good time both in work and daily activities. I appreciate the knowledge shared by Dr. Jimmy Xu and Dr. Shanying Zhu. I have also learned a lot when working with Dr. Chaoyang Jiang and Dr. Masood K. Mustafa. I thank Mr. Deqing Zhai and Mr. Zhen Cao for their generous help. I express my gratitude to Dr. Xianwei Li for his useful suggestions in both study and life. I also appreciate the help and friendship from other research group members. I would also like to acknowledge the research scholarship provided by the SinBerBEST program, which is funded by Singapore's National Research Foundation and led by the University of California, Berkeley (UCB), in collaboration with Singapore universities. Through this scholarship and research opportunity, I am able to work on this interesting project.

I enjoy studying and working with fellow members in the laboratory of NTU. I am happy to work with Mr. Rui Zhao. And I appreciate Dr. Le Zhang for sharing his experience and resources that are useful to my work. I am grateful to Mr. Zhe Wei for accompanying me through tough times. I also appreciate Dr. Zhirong Qiu and Dr. Zai Yang for their suggestions. I am happy to work with many inspiring and humorous colleagues in the laboratory.

Last but not least, I would like to thank my family and friends in supporting my graduate study. I am thankful to my hero, Mr. Benny Hu, for his encouraging words and actions. I am most grateful to my parents for their love and unconditional support. I am thankful to have Miss Wei Peng in my life.

Abstract

Minimizing energy consumption and maintaining occupants' comfort levels are of topical and growing interest as people spend much time inside buildings. To achieve these goals in smart buildings, it is necessary to study the problem of occupant sensing by leveraging machine learning methods to understand occupants based on sensor signals. In this dissertation, we study occupant sensing by answering this question in two aspects: what are the occupants' levels and behaviors?

Existing methods based on feature engineering for occupant sensing are still unsatisfactory. In this thesis, we propose four frameworks based on feature learning to perform occupancy level prediction and occupant behavior recognition. For occupancy level prediction, we propose a framework of neural networks with local receptive fields to process the environmental data, especially those of CO_2 . Real experiments are conducted in a laboratory to evaluate the effectiveness of the proposed framework. Besides, we demonstrate that when the time-domain data is combined with frequency-domain ones, it could provide much valuable information during the feature extraction process to improve the performance of the classification of occupancy levels.

Occupancy levels are the coarse information, but occupants' behaviors are the fine-grained ones. The occupant behavior recognition problem is further divided into simple motion-related activity recognition with inertial measurements and energy-related sound event recognition. For simple motion-related activity recognition based on Inertial Measurement Unit, i.e. accelerometer, gyroscope and magnetometer, we propose a semi-supervised learning framework using Deep Long Short-Term Memory method. As the unlabeled data is pervasively available, the proposed method could enhance performance of the classification of simple motions. Moreover, for lightweight computing purpose, a feature learning framework based on dictionary learning is also proposed to recognize simple motion-related activities based on built-in inertial sensors of smartphones. Real experiments based on our collected inertial sensing data and some public data demonstrate that the

performance of the proposed simple occupant behavior recognition is significantly improved.

As simple behavior alone is not enough to enrich the understanding of occupants and indoor environments, energy-related sound event recognition is also studied for the interactive activities between occupants and appliances based on built-in microphones of smartphones. We propose a dictionary learning framework for the computationally lightweight sound event recognition application on smartphones. Furthermore, to improve the performance based on distinctive representations of recorded sound signals, the feature learning framework based on neural networks with local receptive fields is also proposed to recognize energy-related sound events. Experimental results based on our collected audio dataset and some public dataset demonstrate the effectiveness of the proposed frameworks.

Contents

Acknowledgements	i
Abstract	iii
List of Figures	ix
List of Tables	xiii
Symbols and Acronyms	xv
1 Introduction	1
1.1 Background and Motivation	1
1.2 Major Contributions	4
1.3 Outline of the Thesis	6
2 Literature Review	9
2.1 Occupant Sensing Techniques	9
2.2 Occupancy Prediction	12
2.2.1 Time-of-Survey based Occupancy Prediction	13
2.2.2 Sensor-based Occupancy Prediction	15
2.3 Simple Motion-related Activity Recognition	18
2.3.1 Supervised learning	18
2.3.2 Semi-supervised Learning	22
2.4 Energy-related Sound Event Recognition	23
2.4.1 Low-Level Features	24
2.4.2 Handcrafted High-Level Features	25
2.4.3 Learning-based High-Level Features	25
3 Occupancy Level Prediction with Environmental Sensing Using Feature Learning	27
3.1 Methodology	27
3.2 Experiments and Discussions	33
3.2.1 Data Collection	33
3.2.2 Experimental Setup and Evaluation Criteria	35

3.2.3	Results and Discussions	37
3.2.4	Time and Frequency Domain Data	44
3.3	Summary	46
4	Simple Motion-related Activity Recognition with Inertial Measurements Using Semi-supervised Deep Learning	49
4.1	Methodology	50
4.1.1	Deep Learning for High-Level Features Extraction	51
4.1.2	Generalization based on Randomness	53
4.1.3	Losses and Temporal Ensemble	54
4.2	Experiments and Discussions	58
4.2.1	Dataset	58
4.2.2	Experimental Setup	58
4.2.3	Semi-supervised Learning Comparison	59
4.2.4	Empirical Parameters	61
4.3	Summary	63
5	Simple Motion-related Activity Recognition with Inertial Measurements Using Dictionary Learning	65
5.1	Methodology	65
5.2	Experimental Setup	69
5.2.1	Dataset Collection	69
5.2.2	Feature Extraction and Clustering	70
5.3	Experimental Results	71
5.3.1	Overall Evaluation	71
5.3.2	Accelerometer with and without Gyroscope	74
5.3.3	Statistical Features Versus LLC Feature Learning	76
5.3.4	Limitation	77
5.4	Summary	78
6	Energy-related Sound Event Recognition Using Dictionary Learning	79
6.1	Methodology	79
6.2	Experimental Setup	82
6.2.1	Labeled Dataset	82
6.2.2	Unlabeled Dataset	83
6.3	Experimental Results	85
6.3.1	Comparison between Proposed and Feature-engineering Methods	85
6.3.2	Performance on Web-collected Data	85
6.3.3	Performance on User-recorded Data	86
6.3.4	Discussion on Noise Effect and Limitation	87
6.4	Summary	88

7	Energy-related Sound Event Recognition Using Feature Learning	91
7.1	Methodology	91
7.2	Experiments and Discussions	93
7.2.1	Data Acquisition	94
7.2.2	Implementation Details	95
7.2.3	Experimental Results	96
7.2.4	Sound Detection	99
7.2.5	Robustness against Noise	100
7.2.6	Overlapping Events	100
7.3	Summary	101
8	Conclusion and Future Work	103
8.1	Conclusion	103
8.2	Future Work	106
8.2.1	Integration of Occupant Information for Energy Simulations and Experiments	106
8.2.2	Indoor Localization	106
8.2.3	Other Sensing Techniques	107
8.2.4	Realistic and General Applications	107
A	Codes	109
A.1	Codes of ELM	109
A.2	Codes of RVFL	114
A.3	Codes of Kernel ELM	121
A.4	Codes of KRR	125
	Author's Publications	129
	Bibliography	131

List of Figures

1.1	Occupant sensing resolutions from different perspective	3
1.2	An overview of occupancy and behaviors in a working laboratory	4
2.1	Sensor modality trade-off between occupant sensing performance and intrusiveness	10
2.2	Embedded sensors in a smartphone (Google Nexus 4)	10
2.3	Pattern recognition framework and sensing in the physical world	12
2.4	Occupancy literature review based on two categories	13
2.5	Smartphone Inertial Measurement Unit (IMU) with its coordinate system	20
3.1	Occupancy level with corresponding environmental parameters of CO_2 (ppm), Relative Humidity (%RH), Temperature ($^{\circ}C$) and Air pressure (hPa).	28
3.2	An illustration of 1-Dimensional time series input for local receptive fields convolution with shared weights	31
3.3	Laboratory layout with sensors employed. Cameras were used for ground truth collection.	34
3.4	NRMSE comparison under different window length values.	38
3.5	Accuracy comparison under different window length values.	39
3.6	Detection Rate comparison under different window length values.	40
3.7	Comparison of occupancy level prediction using time-domain frameworks on the testing set	41
3.8	Comparison of occupancy level prediction using time-domain frameworks on the testing set (continued)	42
3.9	Comparison of occupancy level prediction using feature learning with local receptive fields with random weights	45
3.10	Occupancy level prediction of time and frequency domain data based feature learning on the validation sets	46
4.1	Architecture of the proposed temporal ensemble of DLSTM for occupant simple motion-related activity recognition	51
4.2	The DLSTM structure	52
4.3	The LSTM inner structure	52
4.4	The dropout illustration	55
4.5	Comparison with various semi-supervised learning methods	60

4.6	Confusion matrix of our proposed method	61
4.7	Performance of deep neural networks with different number of layers	62
4.8	Deep neural networks performance with different probability of the dropout layers	63
5.1	Architecture of LLC-based framework of smartphone-based occupant simple motion-related activity recognition	66
5.2	Raw acceleration signals of a walking sample	70
5.3	Classification performance on the statistical features of acceleration data. The x-axis indicates the training number in each class.	72
5.4	Classification performance on the statistical features of accelerometer and gyroscope data	72
5.5	Comparison of classification performance with and without the gyroscope data	73
5.6	Comparison of classification performance with and without the gyroscope data (Cont'd)	73
5.7	Comparison of classification performance of acceleration data between statistical features and LLC learning	74
5.8	Comparison of classification performance of acceleration data between statistical features and LLC learning (Cont'd)	75
5.9	Comparison of classification performance of accelerometer and gyroscope data between statistical features and LLC learning	75
5.10	Comparison of classification performance of accelerometer and gyroscope data between statistical features and LLC learning (Cont'd)	76
6.1	The LLC-based framework using unlabeled acoustic data for energy-related sound event recognition. The blue, red and green arrows denote the labeled training, testing and unlabeled data processing procedure, respectively.	80
6.2	The raw acoustic signals of the 8 energy-related sound events	81
6.3	The visualization of the original signals using FFT, spectrum and MFCCs features of the audio file when occupants are working on personal computer	82
6.4	Comparison between LLC-based and Feature engineering-based method using SVM supervised learning	84
6.5	Comparison between LLC-based and Feature engineering-based method using ELM supervised learning	84
6.6	Overall performance in confusion matrix using user-recorded data when combined with extraneous unlabeled data for dictionary learning using SVM. This result is based on the same conditions as in web-collected dataset.	87
6.7	The LLC-based framework using unlabeled acoustic data for different Signal-to-Noise Ratio performances	88

7.1	Architecture of the ELM-LRF for audio classification using random weights and locally dense connections. (a) Input $\mathbf{x}_n$ is the frame-level spectrogram feature with the size of $d \times d$, and output $\hat{\mathbf{Y}}$ is the probabilistic multi-class value; (b) Local receptive fields: K is the number of local receptive fields for feature mappings, and its size is $r \times r$. The convolution filtered result is $\mathbf{C}$ with the size of $(d - r + 1)^2 \times K$; (c) $\mathbf{H}_c$ is the pooling results; (d) β is determined by using ELM. Finally, predicted label is calculated for L classes.	92
7.2	Data visualization using t-SNE. (Better seen in color)	92
7.3	Confusion matrix of proposed architecture on CREATE recorded dataset with 9 classes. The ID is consistent with Table 7.1.	97
7.4	Performance comparison with Additive White Gaussian Noise.	100

List of Tables

2.1	Occupancy prediction based on different sensor	16
2.2	Summary of the typical statistics as basic features	20
2.3	Summary of the typical models as basic classifiers	21
3.1	The division of occupancy levels based on the number of people . .	36
3.2	Comparison of Occupancy level prediction performance on the test- ing set	38
3.3	Occupancy level prediction performance comparison with or without smoothing	40
3.4	Occupancy level prediction performance comparison between time and frequency domain based feature learning on the testing set. (Freq is short for frequency)	45
4.1	Comparison between traditional semi-supervised learning methods and the proposed method	59
5.1	Time and frequency domain feature extraction	71
5.2	Confusion matrix for LLC-based accelerometer and gyroscope features	77
5.3	Leave-One-Subject-Out average classification performance	77
6.1	Index of occupant activity for energy-related sound event recognition	83
6.2	Comparison between LLC-based and feature-engineering based meth- ods using supervised learning algorithms on all labeled data	86
6.3	Overall confusion matrix using web-collected dataset for dictionary learning	86
7.1	Summary of frame-level performance of energy-related testing dataset in terms of frame level on CREATE recorded dataset.	94
7.2	Comparison of classification accuracy based on CREATE recorded dataset. The testing results are reported in the format of average accuracy (its standard variance). The 1-delta and 2-delta denote the 1st-order and 2nd-order difference of MFCCs.	96

7.3	Comparison of classification accuracy in terms of file-level is based on the ESC-50 dataset. The testing results are reported in the format of average accuracy (its standard variance). The 1-delta and 2-delta features denote the 1st-order and 2nd-order difference of MFCCs. CNN-1D and CNN-2D stand for Convolutional Neural Network with 1 and 2 dimensional inputs, respectively.	98
7.4	Comparison of the feature learning structures in audio classification. CNN is the Convolutional Neural Network, and 1D and 2D are the input dimensions. BP is short for Back Propagation. MLP denotes the Multilayer perceptron and RF is short for Random Forest. LRF is short for Local Receptive Fields and NN is short for neural networks.	99

Symbols and Acronyms

Symbols

$\mathbb{R}^n$	the n -dimensional Euclidean space
$\ \cdot\ _2$	the 2-norm of a vector or matrix in Euclidean space
$\odot$	the Hadamard (component-wise) product
$\langle \cdot, \cdot \rangle$	the inner product of two vectors
$diag(\cdot)$	diagonal matrix
x_i	the i -th component of a vector x
$\mathbf{1}$	all-ones column vector with proper dimension

Acronyms

Acc	Accuracy
ANN	Artificial Neural Network
BLSTM	Bidirectional Long Short-Term Memory
BP	Back Propagation
BPTT	Back Propagation Through Time
CNN	Convolutional Neural Networks
CRF	Conditional Random Field
CRVFL	Convolutional Random Vector Functional Link Network
CSI	Channel State Information
DFT	Discrete Fourier Transform
DLSTM	Deep Long Short-Term Memory
DR	Detection Rate
DT	Decision Tree
ELM	Extreme Learning Machine

ELM-LRF	Local Receptive Field based Extreme Learning Machine
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HVAC	Heating, Ventilation and Air-conditioning
IMU	Inertial Measurement Unit
KNN	K Nearest Neighbor
KRR	Kernel Ridge Regression
LDA	Linear Discriminant Analysis
LSTM	Long Short-Term Memory
LLC	Locality-constrained Linear Coding
MFCCs	Mel-Frequency Cepstral Coefficients
NB	Naive Bayes
NLOS	Non-Light-Of-Sight
NMF	Non-negative Matrix Factorization
NRMSE	Normalized Root Mean Square Error (NRMSE)
PID	Proportional-Integral-Derivative
RF	Random Forest
RMSE	Root Mean Square Error
RSSI	Received Signal Strength Indicator
RVFL	Random Vector Functional Link Network
SR	Sparse Representation
SNR	Signal-to-Noise Ratio
STFT	Short Time Fourier Transform
SVM	Support Vector Machine
TEDLSTM	Temporal Ensemble of Deep Long Short-Term Memory

Chapter 1

Introduction

1.1 Background and Motivation

According to the monthly energy review of U.S. Energy Information Administration, the energy consumption of residential and commercial buildings accounts for approximately 40% of the total primary consumption [1]. The energy is mainly used for maintaining a comfortable and healthy indoor environment for occupants. With the ever-increasing population spending most of the time living and working indoors, the percentage of energy consumption by buildings is expected to increase further. Thus, the issue of optimal energy usage by buildings is of significant interest in recent years.

In residential and commercial buildings, the heating, ventilation and air-conditioning (HVAC) systems are one of the most energy-consuming components. Unfortunately, these systems are not yet efficient currently. However, by understanding the occupants and the indoor environment, these systems are capable of working effectively and efficiently [2–4]. To save energy, one must tackle the key problems behind the control and management of the HVAC systems to make them more effective and efficient, for example, by equipping them with the ability to control their settings in response to the occupancy levels and occupant behaviors.

Besides energy saving and human comfort, it is also meaningful to understand occupants' activities for their safety and health. In certain emergency scenario, occupants inside the buildings can evacuate smoothly with some helpful instructions

and the building managements system can effectively manage and understand the whole situations of the building, which ensures the safety of occupants. Further, with a rapidly aging society, it is another big concern to understand the activities of older occupants. For example, fall detection and some alert alarms can be incorporated in a smart way when assisted by a smart building system.

The occupancy level is useful for the building management system. It can be used to save energy by estimating possible energy demands and controlling the HVAC systems. According to some experimental studies and simulations in existing literature, with this elementary occupancy information, energy savings of 10% to 30% have been reported in both simulations and realistic experiments [5–9].

While the occupancy level is the *coarse* information, the occupant behavior is a *fine-grained* one. It has been reported that the occupant’s attitude towards the need for energy conservation has not been effectively improved despite some media campaigns through TV and radio [10]. This is one of many examples that illustrates that occupants tend to take energy usage for granted and are mostly unaware of associated energy wastage and adverse effects on the environment. Hence, it is useful to develop an intelligent system that can identify occupants’ behaviors and use the extracted information to automatically control the building and associated energy management systems. In recent years, some companies have already filed several patents for occupancy pattern estimation, modeling and prediction [11, 12].

Occupants’ behaviors are useful fine-grained information that the building management system can utilize for achieving the goal of reducing energy wastage in modern buildings. Recent studies emphasized the positive effect of occupancy levels and occupant behaviors on energy conservation [13–15]. They pointed out that conservation and wasteful behaviors would have different effects on energy consumption, indicating the necessity and usefulness of understanding occupants’ behaviors. It is believed that the information on occupants’ behaviors can enable the building management system to be more precise and realistic.

Occupant sensing is about the mining of relevant occupants’ information, e.g. occupancy levels and their behaviors, based on physical sensors in modern buildings, and there has been an increasing volume of research in this area. In a recent review, Rafsanjani *et al.* have reviewed various energy-related behaviors in commercial buildings [16]. Melfi *et al.* have also examined occupant sensing from various

perspectives using the temporal, spatial and occupant resolutions [17]. These definitions and resolution levels are simplified and illustrated in Figure 1.1, and they are further explored in this dissertation. The occupancy level is the fundamental element, and the occupant’s location and behavior are the much more detailed ones.

FIGURE 1.1: Occupant sensing resolutions from different perspective

In this thesis, our notion of occupant sensing is restricted to occupancy level prediction and occupant behavior recognition. The latter task is further divided into simple motion-related activity recognition and energy-related sound event recognition. Figure 1.2 illustrates the setup for gathering information on occupancy and behaviors in a working laboratory. From Figure 1.2, with the gathered information on occupancy and activities, one can readily understand the situation in this laboratory. The building management system can then be more effectively operated with the aid of extracted occupant’s information. Although occupant’s location is also an important factor in occupant sensing, it is not studied in this thesis but is highlighted as a future work in Chapter 8.

Currently, most studies on occupant sensing attempt to extract relevant features based on some tricks and heuristics, which are called feature engineering methods. They require domain-expert knowledge to design such features and make the recognition system work. However, if the system requires further improvement for some potential applications, new features may be necessary, and these feature engineering techniques are somewhat not scalable. This is an interesting problem that is

FIGURE 1.2: An overview of occupancy and behaviors in a working laboratory

worth exploring. Features are important in many classification tasks and we aim to learn the features automatically by using effective feature learning approaches to achieve satisfactory recognition performance.

1.2 Major Contributions

In this thesis, four feature learning frameworks are proposed to address the two highlighted issues of occupant sensing. The main contributions of this thesis are summarized as follows.

1. Occupancy Level Prediction

A feature learning framework is proposed to achieve efficient performance on classification of occupancy levels. This feature learning process is non-iterative and thus very fast. Different from traditional methods that are essentially based on feature engineering and only deal with the time series in the temporal aspect, we propose to use the feature learning method to automatically extract better representations based on raw-data inputs in both time and frequency domains. Besides, environmental sensing is used for its non-intrusive property when compared to other sensing techniques. The proposed framework is evaluated to yield superior performance over other traditional methods in a real experiment conducted in a laboratory equipped with environmental sensors.

2. Occupant Behavior Recognition

(a) Simple Motion-related Activity Recognition

- i. A Deep Long Short-Term Memory (DLSTM) semi-supervised learning method with a temporal ensemble is proposed to recognize simple motion-related activities based on inertial measurements. Traditional methods relied on labeled data which is scarce, while the pervasively available unlabeled data cannot be leveraged. Besides, simple statistics are low-level features that cannot significantly improve the performance of the recognition system. We make use of DLSTM to deal with unlabeled data based on deep learning methods that are novel approaches in extracting the local dependency and temporal scale-invariant representations as high-level features. The proposed method yields superior results over other state-of-the-art semi-supervised methods.
- ii. A dictionary learning framework is proposed for high-level feature extraction for simple-motion activity recognition. This framework based on Locality-constrained Linear Coding (LLC) is computationally lightweight when compared to other dictionary methods, e.g. Sparse Coding that is demonstrated to be satisfactory in the activity recognition. The improvement is achieved by using encoded representations as high-level features instead of the usual representations from feature engineering. We also establish a dataset for

simple motion-related activity recognition to demonstrate the efficiency of the proposed framework.

(b) Energy-related Sound Event Recognition

- i. The dictionary learning framework is applied to energy-related sound event recognition. This framework can learn high-level features by leveraging unlabeled data that most traditional methods cannot handle. The effectiveness of the approach is demonstrated with our dataset that is collected by using the microphone embedded in a smartphone. We focus on the sound recognition of activated electrical appliances and other utility resources commonly found in offices and/or laboratories. This dataset can be accessed by the public¹.
- ii. A feature learning framework based on neural networks with local receptive fields is proposed for the recognition of energy-related sound events. Traditional methods are based on the well-extracted Mel Frequency Cepstral Coefficients (MFCCs) which are originally designed for speech recognition but not sound recognition. The proposed framework learn high-level features based on the raw-data inputs in both the time and frequency domains. Through evaluations using both the public dataset and our smartphone recorded dataset, the proposed framework outperforms other state-of-the-art methods. Furthermore, the proposed non-iterative framework is significantly more efficient than deep learning methods that require the tuning of many hyperparameters such as the learning rate, the number of epochs, and others.

1.3 Outline of the Thesis

The remainder of the thesis is organized as follows:

- In Chapter 2, existing literature on occupancy prediction and occupant behavior recognition is reviewed and discussed. First, various occupant sensing

¹https://drive.google.com/file/d/0B6KX6IV_47RyUnZMcFB2YnR3UE0/view?usp=sharing

techniques are examined to search for an effective and non-intrusive solution. This is followed by a review on occupancy prediction. Then, occupant behavior recognition is discussed from the two aspects of inertial sensor-based simple motion-related activity recognition and microphone-based energy-related sound event recognition.

- In Chapter 3, a feature learning framework is proposed to classify occupancy levels based on chemosensor data. With raw environmental-sensing data, a neural network based feature learning framework is presented. It incorporates both the time-domain and frequency-domain information to improve the system's performance.
- In Chapter 4, a semi-supervised deep ensemble learning method is proposed to recognize simple motion-related activities of occupants based on the public dataset gathered by the built-in IMU of smartphones. We present how unlabeled data can be leveraged to enhance the performance of the simple motion-related activity recognition system.
- In Chapter 5, a dictionary learning framework is proposed to recognize simple motion-related activities of occupants based on information gathered by the built-in IMU of smartphones. Real experimental results for the simple activity recognition are also presented to demonstrate the effectiveness of this smartphone enabled method.
- In Chapter 6, a dictionary learning framework is applied to the energy-related sound event recognition problem. Real experimental results are also presented to demonstrate the framework's effectiveness.
- In Chapter 7, a feature learning framework based on neural networks with local receptive fields is presented for energy-related sound event recognition. The framework's effectiveness is demonstrated through our smartphone-collected dataset and a public dataset.
- In Chapter 8, conclusions are drawn and some ideas for future works are presented.

Chapter 2

Literature Review

We begin this chapter with a discussion on the sensing techniques. Then, existing works on occupant sensing in terms of occupancy and behavior recognition techniques using different sensing modalities are reviewed.

2.1 Occupant Sensing Techniques

Occupant sensing is based on information gathered by the various sensors, ranging from accurate to imprecise, intrusive to non-intrusive. For different sensor modality, we summarize the pros and cons of different type of sensors, especially with regards to the accuracy for occupant sensing and intrusiveness to users. In a survey study, Teixeira *et al.* discussed human sensing using many sensor modalities [18]. Specifically, Alessandra *et al.* focused on building management system with different sensing techniques [19].

The different sensors in terms of a trade-off between performance, i.e. accuracy in mining information, and intrusiveness, i.e. interference to privacy, are analyzed herein. The comparisons of PIR, microphone, smartphone, camera, inertial sensors as well as chemosensors (CO_2 and humidity sensors) are shown in Figure 2.1. Among these sensors, smartphones are smart devices with many sensors integrated into them because of commercial values. Some typical sensors in the Google Nexus 4 smartphone are as illustrated in Figure 2.2. The chemosensors are thermometer, humidity sensor and barometer. The inertial sensors, called Inertial Measurement

FIGURE 2.1: Sensor modality trade-off between occupant sensing performance and intrusiveness

FIGURE 2.2: Embedded sensors in a smartphone (Google Nexus 4)

Unit (IMU), are accelerometer, gyroscope and magnetometer. Most of built-in sensors of smartphones are designed for some applications such as games and interactions, thereby reducing the resolution of performance when embedded in the off-the-shelf smart devices. Many applications based on smartphones are interesting, and wide ranging from mobile sensing to human computer interaction design. Since we mainly focus on occupancy and behavior recognition for our occupant sensing application, we therefore investigate only those sensors relevant to these two aspects.

When examining the deployment of sensors in smart buildings, one extreme example is that of the National Taiwan University project [14]. Many sensors are deployed in the home as a laboratory testbed. Although these sensors, such as touch sensors (the 3-D accelerometer, reed switch and mercury switch), microphone sensors, light sensors and the camera, are able to provide abundant information, the installation and the retrofitting process for a normal room is impractical. In addition, the camera may pose a concern for occupant's privacy. Clearly, accuracy as well as intrusiveness are two important factors when considering the sensor modality.

As a result, we aim to improve the recognition performance while somewhat reducing the intrusiveness. Chemosensors and built-in sensors of smartphones are suitable candidates and we mainly study these sensors in this thesis. Although occupant sensing can leverage on a lot of sensor modalities, we use the typical smartphone for its pervasiveness and potential in mobile computing [20]. When studying the recognition of occupant activities, we assumed that every occupant carry one smartphone.

After data collection via smart devices, the streams of data are to be analyzed to extract the desired information. In pattern recognition, the traditional framework is as shown in Figure 2.3. Most of the existing works have focused on feature extraction and classifier training task. In this thesis, we mainly focus on the feature extraction part. And all the signals studied are time-series data.

On the problem of feature extraction and classifier training, many works have been studied in extensive areas. Among these works, artificial neural networks are powerful tools to achieve significant performance. For example, the randomized neural network is well-studied in some surveys in comprehensive ways with abundant

benchmark comparisons [21] [22] [23] [24]. Specifically, these randomized neural networks are quite useful not only in time-series recognition problems, e.g. energy power prediction [25], but also in other image and video recognition problems [26]. We shall discuss these methodologies in details in the subsequent chapters. In the following subsections, we review the existing literature on occupant sensing, starting with occupancy prediction.

FIGURE 2.3: Pattern recognition framework and sensing in the physical world

2.2 Occupancy Prediction

When it comes to occupancy, many issues related to prediction and estimation have been studied in the literature. The problem of estimation is usually based on the sensing data to find the correlation between the objective occupancy and the environmental parameters. Some works on occupancy are only concerned with two states of presence and absence, i.e. binary classification. Other works on occupancy are more detailed to certain occupancy level or the exact number, which we consider as prediction or estimation here. Most of these methods are data driven, following guidelines in the areas of data mining and machine learning research.

The literature on occupancy prediction is summarized by classifying them into two categories based on different data: time-of-survey data and sensing data. The time-of-survey data is fundamental to the occupancy prediction problem. It is

mainly collected using the camera or through recording manually with much labor. While the camera is also a sensor, it is different from other sensors due to its specific property of carrying much information in the visual form. Closed-circuit television may pose some privacy concern to the occupants, and the object counting still remains a challenging problem if the camera is operating under weak lighting conditions and other practical issues. For example, a mirror can cause interference when counting the number of occupants. As a result, it would be better to use this data as ground truth. Other sensors are not as informative as the camera, but they can still reveal the hidden correlation between occupancy and environmental factors if processed properly. Occupancy prediction can be based on either time-of-survey or sensing data as shown in Figure 2.4.

FIGURE 2.4: Occupancy literature review based on two categories

2.2.1 Time-of-Survey based Occupancy Prediction

Occupancy modeling is based on historical data with the probabilistic formulation and mathematical derivation. Occupancy modeling is concerned with the study of the regular hidden pattern of the occupancy based on some common features such as the schedule on weekdays, thereby relying on the time-of-survey data. First of all, we have to obtain the time-of-survey data about the number of occupants in a room or a zone. Both modeling and data mining methods are capable of solving these specific problems. We first introduce occupancy modeling, then followed by data mining methods on time-of-survey data.

In the review paper, Rafsanjani *et al.* have given a summary of agent based occupancy modeling [16]. The agent based model is considered to be one of the most

popular methods to model the occupancy based on historical occupancy data. One of its drawbacks in modeling the occupancy is that this historical data as ground truths is somewhat rare and laborious to obtain, e.g. through recording by cameras. Page *et al.* have proposed an approach to model the probability of occupancy presence using an inhomogeneous Markov model [27], but the time steps in their transition matrix lead to high computing time in establishing this matrix. Furthermore, Liao *et al.* have proposed a stochastic approach using an agent-based model that is suitable for any zone and any occupant [28]. Wang *et al.* have also proposed the homogeneous Markov model to describe the occupancy level of offices by considering their entering and exiting patterns [29]. By considering the computation time in the inhomogeneous Markov model, Chen *et al.* have modeled the transition between zones for occupants in a time step of a minute, which also solves the problem of multiple-zone multiple-occupant prediction in a fast manner [30, 31]. Erickson *et al.* have compared agent based modeling with multivariate Gaussian model, using real time occupancy in laboratories and offices [32]. Through training on the individual camera data, an occupant is considered as an agent, and the trajectory of an occupant from one zone to another in a time stamp is a sample of the information for this agent. The agent based model can also be used for energy modeling simulation. For example, Lee *et al.* have used an agent based modeling approach to estimate the occupants' behaviors and combined with the Predicted Mean Vote model for energy simulation in the EnergyPlus [33]. Li *et al.* have applied agent based model for occupancy density estimation in the simulation of HVAC system [34]. Azar *et al.* have also employed the agent based model to understand the interactions of the individual occupant who is regarded as agent [35].

Data mining techniques can also be applied to analyze the historical data for the modeling purpose. For the regular patterns of occupancy in the commercial building scenarios, Chen *et al.* have compared various occupancy modeling with data mining techniques [36]. The occupancy modeling methods introduced are the inhomogeneous Markov model and multivariate Gaussian model, whereas the data mining methods include autoregressive integrated moving average, artificial neural network and support vector regression. They found that the performance of model-based occupancy modeling is limited whereas those of data mining approaches are much better.

The historical data is difficult to obtain for occupancy modeling and mining approaches. The sensing data is much easier to obtain than the time-of-survey one, and are effective solutions as well. The methods based on time-of-survey data tend to require large scale historical data, but those based on sensing data do not have this requirement. The sensing methods can indeed enrich the understanding of occupancy prediction both from the historical data as ground truths and the sensing modality as the physical monitor.

2.2.2 Sensor-based Occupancy Prediction

A summary of occupancy prediction based on different sensors is presented in Table 2.1.

From a review of occupancy sensing system, different sensing techniques have pros and cons in terms of its applications to buildings [52]. Liu *et al.* have proposed the wireless sensor networks of PIR sensors to detect the occupancy using the Hidden Markov Model (HMM) [37]. Although the occupied and vacant state can be detected precisely, the output results are limited in their binary form. Li *et al.* have applied the K-Nearest Neighbors (KNN) on the Received Signal Strength Index (RSSI) of RFIDs to build an occupancy detection system for individuals [38]. However, every occupant has to wear an active RFID tag in this system, which is somewhat intrusive. Uziel *et al.* have designed acoustic sensors to capture the occupancy levels based on machine learning algorithms, i.e. the Gaussian Mixture Model (GMM) and HMM [39] but the sensors are sensitive to the noise that occupants would make. Erickson *et al.* have designed a system called POEM based on camera to supervise the occupancy [7]. However, the camera suffers from the illumination factor and cannot be extended to the rooms that call for privacy such as bathrooms and bedrooms.

Besides the somewhat intrusive sensing techniques, other modalities that are friendly to users have also been investigated. Zhao *et al.* have combined many sensors, for example, the PIR, keyboards and mouse sensors, door sensors as well as chair sensors etc. as the virtual occupancy sensor to detect the occupied or vacant state in the room and work zone with the Bayesian belief network [40]. Although these sensors are powerful to get accurate detection rate, this combination is inflexible

TABLE 2.1: Occupancy prediction based on different sensor

Sensors	Literature	Pros	Cons
PIR	Liu <i>et al.</i> [37]	precise in detection	coarse information: only binary output
RFID	Li <i>et al.</i> [38]	multiple occupants' state tracking	intrusive: every occupant has to carry tags
Acoustic	Uziel <i>et al.</i> [39]	non-intrusive	occupancy levels are sensitive to noise in audio
Camera	Erickson <i>et al.</i> [7]	accurate	pose concern to privacy and easily affected by illumination
Virtual occupancy sensors	Zhao <i>et al.</i> [40]	accurate and non-intrusive	labor intensive and unable to count people
Power meter	Jin <i>et al.</i> [41] [42], Zhao <i>et al.</i> [43]	non-intrusive	coarse information: only binary output
Routers	Depatla <i>et al.</i> [44]	solving NLOS and scattering effects	limited positions of routers in dynamic environment
WiFi	Li <i>et al.</i> [45]	non-intrusive using existing access point	interference by dynamic environments
CO ₂	Jin <i>et al.</i> [46], Ebadat <i>et al.</i> [2], Jiang <i>et al.</i> [47]	non-intrusive	slow response
Chemosensor	Masood <i>et al.</i> [48]	redundancy and non-intrusive	hard to analyze signals
Combined	Dong <i>et al.</i> [49] [50], Zikos <i>et al.</i> [51]	accurate	complicated and expensive

and limited in terms of occupancy levels. Jin *et al.* have also proposed to understand the occupancy with power meters in a room in a semi-supervised manner. These power meters are also termed the virtual occupancy sensing as it is non-intrusive to occupants in a house but this method is only limited to the detection of presence and absent states. Zhao *et al.* also investigated data mining methods from the power consumption data with simulation on EnergyPlus by leveraging the occupancy information [43]. Depatla *et al.* have proposed to analyze the RSSI between routers to count the occupancy by tackling the Non-Line-Of-Sight (NLOS) and scattering effects problems [44]. Li *et al.* have proposed to use the existing routers and smartphones in the rooms and applied the Wiener filter and Newton

interpolation to count the number of users inside the room [45]. However, their results are not tolerant to the dynamic environment condition especially changes to furniture.

Among the non-intrusive sensors, the chemosensors are one of the popular sensing modality. Jin *et al.* [46] and Ebadat *et al.* [2] have used CO_2 sensors to find the relationship between this signal and occupancy in a non-intrusive manner as the CO_2 sensor would not interfere with the occupant's work and privacy. However, according to the review [52], the CO_2 signal would have a slow response. With other environmental parameters, e.g. temperature, humidity and air pressure, Masood *et al.* have focused on these signals with feature selection techniques to further improve the occupancy prediction performance. Moreover, integrated sensor testbed including CO_2 has been studied by Dong *et al.* [49, 50] with accurate performance in occupancy prediction but the system is complicated and inflexible to transfer to another building. As discussed in the investigation of the integrated testbed in the CMU lab [50], the environmental parameters are the most relevant features for studying the occupancy level. Jiang *et al.* also investigated these useful signals with a light and efficient algorithm of weight scale Extreme Learning Machine (ELM) to achieve better results [47]. From the comparison between different occupancy sensing modalities, chemosensors are suitable candidates with their non-intrusive advantages and their extendability to other rooms.

The studies on the use of chemosensors for occupancy prediction are based on different models. For example, HMM is a reliable approach to provide a stable output of the occupancy estimation [50] and is extensively applied in this area. With the probabilistic graphical model, the regular pattern of the occupancy in the weekday can be predicted accurately. Furthermore, Conditional Random Field (CRF) is superior to HMM [51] as HMM has more assumptions than CRF. In terms of machine learning approaches, Yang *et al.* have investigated different machine learning classifier models for the occupancy prediction in a systematic manner [53]. They have compared Artificial Neural Networks (ANN), Support Vector Machine (SVM), Naïve Bayesian (NB), K-Nearest-Neighbor (KNN), Decision Tree (DT) and tree-augmented naïve Bayes network, and reported that DT can provide the best results. Candanedo *et al.* have also compared some of the machine learning algorithms involving Classification and Regression Trees (CART), Random Forest (RF) and Linear Discriminant Analysis (LDA), and found that LDA can give a

satisfactory performance [54]. Masood and Jiang *et al.* have applied ELM to predict the occupancy in a quick manner and achieve better results [47, 48]. Most of the existing works depend on the well-extracted features despite the different classification models they chose. And these features are extracted based on experience during experiments.

Recently, feature learning is extensively studied in the pattern recognition area such as image, speech and natural language processing. The traditional methods for image classification are based on the Scale Invariant Feature Transform features [55]. Speech recognition is basically based on the Mel Frequency Cepstral Coefficients (MFCCs) [15]. As dictionary learning and deep learning have shown their advantages in extracting high-level features, feature learning is indeed a promising approach in the occupant sensing area. However, no prior work on occupancy prediction using feature learning methods has been reported. Thus, we aim to study how one can obtain the suitable features to distinguish occupancy levels or occupancy numbers, especially with the chemosensors data.

2.3 Simple Motion-related Activity Recognition

2.3.1 Supervised learning

According to a recent survey on human activity recognition using body-worn inertial sensors [56], this problem is studied extensively in applications of human computer interaction, mobile and ubiquitous computing. A prevalent method to study human activity is to use images and videos. However, vision is complicated and sometimes the computer cannot understand the scenario under some poor conditions. As a result, body-worn inertial sensors become one of the most feasible sensors to study human activities. As the number of wearable devices and smartphones is increasing in the era of Internet of things, inertial sensors are pervasive and readily available as the commercial off-the-shelf products. Inertial sensors are capable of recording the physical motion of the users.

In 2004, Bao *et al.* investigated the activity of daily living based on acceleration data from user-annotated records [57]. The data acquisition process requires users

to carry devices with the Micro electromechanical systems (MEMS), and the happening time of an event and its category should be annotated during or after the action. Ward *et al.* have investigated the activity of assembly workers based on accelerometers [58]. The time-series data from inertial sensors is basically acceleration, angular acceleration and direction from the accelerometer, gyroscope and magnetometer, respectively. For temporal data segmentation, the sliding window is a common tool to deal with the shortage of dimensions in each time step [59]. Another preprocessing method is to calculate the magnitude of the signals with an emphasis on its energy [60]. With ground truth data as training samples, instances from the raw data are processed by feature extraction process. The classification of activities include simple human physical actions and the ubiquitous interactive environment. For example, walking and running are simple actions and taking a bus and an elevator are the actions that users may perform in their daily life. In order to enhance the development of human activity recognition, many datasets are open source and freely accessible [61–63].

The simple motion-related activity recognition is investigated mostly on smartphones as they could monitor occupants' simple actions by accessing the built-in motion sensors. According to a survey of the human activity recognition on smartphones [64], many perspectives of this problem have been studied. For example, different brands of smartphone devices have different operating systems, e.g. Symbian, Android and iOS. Some of the recognition algorithms are off-line whereas others can directly run online with some resource consumption analysis in terms of CPU, memory and battery usages. However, when considering the use of a smartphone as a motion sensor, some practical problems should be tackled. These problems are the orientations and on-body placements of smartphones. The other IMUs are accurate and should be fixed to certain on-body placement, so they can easily handle and avoid the orientation problem. For example, the MotionNode used in the dataset [63] and experiment [59] are fixed to the waist of the subjects. Furthermore, These sensors are more accurate than the built-in IMUs of smartphones that are originally designed for screen rotation and some games which do not require the high resolution.

The coordinate system of built-in IMU of smartphones is as illustrated in Figure 2.5. Some of the existing works have chosen the magnitude of the acceleration data to

TABLE 2.2: Summary of the typical statistics as basic features

Domain	Features
Time	maximum
	minimum
	magnitude
	mean
	coefficient of variance
	standard deviation
	root mean square
	correlation between axis
	median
	signal magnitude range
	interquartile range
Frequency	maximum
	minimum
	FFT energy
	skewness
	kurtosis
	mean
	spectral entropy
	sub-band energy ratio

tackle the orientation problem [60, 65]. In fact, Cleland *et al.* have studied the on-body positions of the accelerometer, finding that among the so many placements, e.g. wrist, chest, lower back, thigh and hip, the best placement is in the lower torso because many simple physical actions are dictated by the lower part of the human body. Henpraserttae *et al.* have proposed a projection based approach to solve the problems of the device orientation and on-body location [66].

FIGURE 2.5: Smartphone Inertial Measurement Unit (IMU) with its coordinate system

TABLE 2.3: Summary of the typical models as basic classifiers

Literatures	Approaches	Activities	#Classes
Bao <i>et al.</i> [57]	Decision tree (C4.5)	daily activities	10
Ward <i>et al.</i> [58]	HMM/LDA	assembly jobs	9
Zhang <i>et al.</i> [59]	Sparse Representation	physical motions	9
Bhattacharya <i>et al.</i> [60]	Sparse Coding	transportation modes	6
Anguita <i>et al.</i> [61]	SVM	physical motions	6
Henpraserttae <i>et al.</i> [66]	KNN	physical motions	6
Lane <i>et al.</i> [67]	Naive Bayes	transportation modes	4
Kim <i>et al.</i> [68]	Hierarchical SVM	physical motions	5
Blanke <i>et al.</i> [69]	CRF	composite physical activities	10

Regardless of the practical problems of using inertial measurement units for human activity recognition, feature extraction and classifier training are the most important steps in so many existing works. The typical methods to extract features are based on statistics. The time and frequency domain features are summarized in Table 2.2. Normally, the maximum, minimum and mean are calculated both in time and frequency domains. Frequency-domain data is based on Fast Fourier Transform (FFT). Besides, some statistics are calculated separately in each domain. For example, the signal magnitude range and interquartile range are used in the time domain whereas energy is used in the frequency domain. These features are designed based on experience and turned out to be more effective than the raw data in the traditional classifier training process. The extensive classifier models are applied based on extracted features.

We have also summarized the models used in the literature, which is tabulated in Table 2.3. Bao *et al.* proposed to use DT to solve the acceleration classification problem [57]. Ward *et al.* then proposed HMM and LDA for microphone and acceleration classification, respectively, to identify actions of the workers in the assembly line [58]. Zhang *et al.* applied a sparse representation method to classify the physical motions, which is somewhat accurate because of the high resolution of the MotionNode sensors [59]. Furthermore, the sparse coding method is applied directly on the magnitude of the acceleration data for the classification of transportation modes, e.g. taking a bus or a train [60]. Lane *et al.* also investigated the transportation modes such as riding a bike or taking a vehicle based on the Naive Bayes classifier [67]. Anguita *et al.* proposed to use SVM as the baseline on their public dataset of smartphone acceleration data [61]. Zhang *et al.* proposed to use feature selection method including wrapper based method with

SVM as classifier for nine types of physical motions [70]. When considering the orientation-independent problem, Henpraserttae *et al.* presented the use of KNN with a projection method to classify 6 physical motions based on the iPhone [66]. Kim *et al.* proposed a hierarchical SVM method to classify the physical motions so as to estimate the metabolic rate in their energy expenditure model for smart buildings and healthcare applications [68]. Blanke *et al.* proposed the use of CRF to transfer the knowledge learned in the physical motions to other composite daily working activities, e.g. installing a bookshelf [69].

Recently, as feature learning has developed quickly, some researchers have tried to investigate how one can extract the features automatically. The first investigation is the use of dictionary learning. For example, Vollmer *et al.* proposed to use shift-invariant sparse coding to find suitable features for activity recognition [71]. Bhattacharya *et al.* also proposed sparse coding to learn features for transportation activity recognition based on the magnitude of the raw data [60]. Another attempt is to investigate deep learning methods for learning the distinguishable features. For example, Plotz *et al.* proposed Restricted Boltzmann Machine (RBM) based on the raw data and the empirical cumulative density function, and found that the learned features outperformed the original statistical features both in time and frequency domains [72]. Zeng *et al.* proposed partial weighted Convolutional Neural Networks (CNN) on the raw data to learn the high-level features because CNN can extract features automatically by capturing the local dependency and scale invariance from the signals [73]. Ordonez *et al.* proposed a combination of convolution and Long Short-Term Memory (LSTM) recurrent neural network to extract high-level features based on different modalities of motion sensors [74]. The advanced deep learning framework outperformed the partial CNN methods both on OPPORTUNITY [62] and Skoda dataset [75]. Besides, Yang *et al.* also proposed a deep method for the multichannel time series signals in comparison with deep belief network [76]. Alsheikh *et al.* proposed a spectrogram based hybrid deep learning and HMM approach and reported better results on some datasets [77].

2.3.2 Semi-supervised Learning

As the supervised learning methods rely heavily on annotations, it is sometimes difficult to obtain labeled data. Because it is time-consuming, most of the occupants

are not willing to record their actions every time. Semi-supervised learning and multi-instance learning are required for exploring the human activity recognition problem. Stikic *et al.* investigated the weakly supervised learning, i.e. multi-instance learning, and semi-supervised learning [78] in human activity recognition. They first studied self-training [79] and co-training [80] methods to deal with unlabeled data [81]. They then proposed a multi-graph label propagation method in the semi-supervised learning task [82]. Guan *et al.* proposed an auto-regressive HMM for multi-instance learning in an efficient manner based on dynamic programming [83]. Yao *et al.* proposed a shared dictionary approach for feature learning in a semi-supervised manner by leveraging unlabeled data with different constraints as regularization [84]. Rokni *et al.* recently proposed a framework of synchronous dynamic view learning for classifier training in an autonomous manner to eliminate the annotation process that is time-consuming and labor intensive [85].

2.4 Energy-related Sound Event Recognition

Occupants of a room or an office do require many resources, and they frequently interact with the resource-related facilities. These facilities are the typical electrical appliances and other equipment related to energy and resource utilization. When these facilities are activated by occupants, they will release some signature sounds. Machine learning techniques can be deployed to identify the activated facility by processing the recorded sound. This is an artificial intelligent task as discussed on the concept of machine hearing for understanding sound events [86]. The related problem is bird sound classification as studied in the previous work by Stowell *et al.* [87]. The focus on automatic speech recognition is to recognize the language itself, whereas sound event recognition is on scene understanding, active activity and acoustic surveillance based on audios from the microphone. Instead of computational auditory scene analysis which is in the physiology and neuroscience area, sound classification in smart buildings belongs to the domain of signal processing and machine learning. In sound event classification, Englert *et al.* proposed to use the built-in microphone of smartphones to classify the electrical appliances when people are using them [15]. Dennis *et al.* summarized sound event recognition applications, indicating that non-speech recognition is a meaningful research problem [88].

As the study of non-speech audio application is gaining momentum, more and more datasets are set up and competitions are held to promote this research area. In 2007, a dataset from CLEAR was released during a workshop for the first time to investigate sound event, activity and relationship classifications [89]. CLEAR is short for classification of events, activities and relationships. Later in 2013, a similar competition called DCASE was held and summarized in the review paper [90]. DCASE is the abbreviation of detection and classification of acoustic scenes and events. Then, two consecutive competitions called DCASE [91] were held to further understand this problem. However, these datasets are not large scale and include extensive scenes and they are sometimes not all fitted for smart building applications. As more large scale public datasets are released recently, e.g. Google Audioset [92] and Youtube8M with audios [93], the understanding of the sound recognition would be explored in a meaningful manner similar to the Imagenet in the computer vision problem [94].

The traditional methods to recognize sound events are based on features and classifier models. Feature extraction starts from low-level features to high-level features. The low-level features are simply based on temporal or spectrogram data. With the use of window framing, MFCCs that are originally designed in traditional speech recognition [95] are somewhat useful in sound event recognition. The traditional methods to deal with audios are to first preprocess the whole audio into window frames with a certain percentage of overlapping. Then, one needs to extract the low-level features, e.g. MFCCs. Later, HMM can be used to model the auditory and conduct the classification performance. Other related classifiers are GMM, ANN and SVM.

2.4.1 Low-Level Features

In terms of audio features, many characteristics studied are the low-level ones. Pitch, loudness, duration and timbre are some typical features that could describe the acoustic signals based on simple statistics [88]. However, the temporal characteristics alone are not enough to deal with the complicated acoustic signals, one usually has to use the frequency domain for spectrogram analysis. For example, the MFCCs are the typical features obtained by first applying Fourier transform, then the Mel banks filtering, followed by discrete cosine transform eventually.

Zero crossing rate, band energy, bandwidth and power spectrum are some traditional features. Zero crossing rate and bandwidth are also used in some mobile computing applications of sound classification [96, 97]. The other related low-level features are spectral flux, roll-off, entropy and phase deviation. The Delta MFCCs and Double Delta MFCCs are applied based on MFCCs as discussed in the previous work by Englert et al. [15]. Some related features, e.g. Gammatone Cepstral Coefficients, use the gammatone filtering instead of Mel filtering and this spectrogram are not Mel-based but cochlea-based [98, 99].

2.4.2 Handcrafted High-Level Features

Further exploration of high-level features of audios is based on the two-dimensional time frequency as an image. Normally, the audios are time-series signals that only have little information in the time domain at each sampling step. Thanks to the Short Time Fourier Transform (STFT), which is a typical method to create these images for each temporal windowed frames, the one-dimensional time-series signals can be transformed into an image of the spectrogram. With the experience and knowledge learned in the image processing area, some researchers are inspired to design high-level features.

These high-level features are based on auditory image processing. For example, the Histogram of Gradients have been applied for audio scene classification [100] although it is originally designed for computer vision [101]. Walters has also proposed the Auditory Image Model and based on it, derived the Stabilized Auditory Image for sound classification [102]. Furthermore, Dennis *et al.* proposed the Spectrogram Image Features to process the time-frequency representation of auditory images [103]. Another approach inspired from image processing is the generalized Hough transform, which they named it local spectrogram features for overlapping sound classification [104].

2.4.3 Learning-based High-Level Features

Similar to bag-of-visual-word, the dictionary learning methods are also studied for audio recognition. Chu *et al.* proposed the Time-Frequency Representation and using matching pursuit for selected features in the dictionary [105]. Heittola *et al.*

proposed the Non-negative Matrix Factorization approach to find a suitable basis for better features and HMM is applied in the classifier model [106]. And the sparse coding method is also applied to sound event recognition [107]. The K-means is reported to improve the classification results in the biological sound recognition [87] and urban sound event classification [108]. The disadvantage of the dictionary learning methods is that during the clustering process, the time needed to solve the optimization problem is rather long.

Deep learning methods have also been studied for sound event recognition. Hertel *et al.* proposed a 4-layer one-dimensional CNN using the frequency domain input for the DCASE 2016 competition [109]. Bidirectional Long Short-Term Memory (BLSTM) with Denoising Autoencoder is applied for sound event detection [110] and BLSTM with data augmentation is studied for extracting high-level features in polyphonic sound classification [111]. As LSTM can access the context with a persistence of past content while processing the present task, they applied LSTM and even BLSTM to learn the features in two directions. The DNN is applied based on the Stabilized Auditory Image and Spectrogram Image Features [112].

Spiking neural network (SNN) is also studied for the sound event classification problem. The neurons in spiking neural networks do not depend on the activation functions any more but on the spiking input and output instead. It now works effectively in hardware. It may provide a satisfactory application to the occupant behavior recognition problems. For example, Dennis *et al.* has investigated the usage of SNN to classify different sound events in a robust way [113]. The SNN is the third generation neural network that is inspired from biological phenomenon as studied in the work by Sumit *et al.* [114]. It is less abstract than the traditional nonlinear activation functions in the previous artificial neural networks. Consequently, it is easy to design and implement in hardware as it is based on the natural biological schemes.

Chapter 3

Occupancy Level Prediction with Environmental Sensing Using Feature Learning

In this chapter, we present our framework for occupancy level prediction. The problem is to identify the level of occupancy in a room or office. This is of practical importance for the control and management of lighting and HVAC systems as studied in previous works [115, 116]. Traditionally, the occupancy information is mined by somewhat intrusive sensing techniques. Our interest here is to study the use of environmental sensing, i.e. sensing via the chemosensors, to explore the relationship between occupancy levels and environmental parameters such as CO_2 , temperature, humidity and air pressure. Besides, current methods are normally based on statistics as features [50], but we explore this problem by using feature learning methods.

3.1 Methodology

We begin with the mathematical formulation of occupancy level recognition using environmental sensing. The relationship between the number of occupants and the environmental signals is shown in Figure 3.1. These temporal signals are closely correlated, especially the trend of the CO_2 signals which has a strong correlation with the occupancy levels, albeit with a delay.

FIGURE 3.1: Occupancy level with corresponding environmental parameters of CO_2 (ppm), Relative Humidity (%RH), Temperature ($^{\circ}C$) and Air pressure (hPa).

We define the time-series signals as

$$\mathbf{x} = [x_1, x_2, \dots, x_t, \dots, x_T]^T \quad (3.1)$$

where $\top$ denotes the transpose of a vector, and $t = 1, 2, \dots, T$ denote the discrete time steps. The CO_2 concentration, temperature, relative humidity and air pressure can be described by this T -length sequence.

The time domain raw data has been used to extract features empirically in the previous works [48] [50]. It is introduced here in details for the purpose of comparison with our proposed feature learning method which does not need this feature engineering technique at all. The 1st-order and 2nd-order differences are the typical features used to separate different levels of occupancy. Moreover, the moving average within a time period and the time shift difference are also meaningful features for occupancy prediction. These engineered features are called proportional-integral-derivative (PID) features according to the term defined in the work by Jiang et al. [47]. The calculations of these features in the time domain are based on the following formulae:

$$\begin{aligned}
tfd_t &= x_t - x_{t-1} \\
tsd_t &= tfd_t - tfd_{t-1} \\
tds_t &= x_t - x_{t-s} \\
tag_t &= \frac{1}{N} \sum_{i=t}^{t+N} x_i
\end{aligned} \tag{3.2}$$

where tfd_t denotes the time-domain first-order difference at time t , tsd_t denotes the time-domain second-order difference based on the first feature, tds_t signifies the time-shift difference with s time delay, and tag_t is the moving average within a window of N time steps. These terms can be concatenated into one vector as the features for every instance, i.e.

$$\mathbf{Xfeat} = [\mathbf{tfd}; \mathbf{tsd}; \mathbf{tds}; \mathbf{tag}] \tag{3.3}$$

These feature engineering extraction methods can be applied to all the environmental signals, and the four sensors mentioned above can be extracted and concatenated together.

Normally, these time domain features are fed into the classifier for training. These features are designed with some tricks and heuristics, which requires some domain-expert knowledge. However, the proposed framework presented in this chapter can accomplish the feature extraction task with minimal expert knowledge by using feature learning. Furthermore, the existing feature extraction methods processed the time domain data only. The frequency domain data, which can be processed by using Fourier transform, has been demonstrated to be very valuable in numerous problems such as image and audio processing. We demonstrate that the occupancy classification performance can indeed be significantly improved when the time domain data is integrated with the frequency domain data. The frequency domain data is obtained by using the Discrete Fourier Transform (DFT):

$$X_k = \sum_{t=0}^{T-1} x_t e^{-2\pi jkt/T} \quad (k = 0, \dots, T-1) \tag{3.4}$$

where $j = \sqrt{-1}$ is the unit of imaginary number. The magnitude of the frequency domain coefficients is given by:

$$|X_k| = \sqrt{Re(X_k)^2 + Im(X_k)^2} \quad (3.5)$$

where $Re(.)$ and $Im(.)$ are the real and imaginary parts, respectively. After obtaining the frequency domain data, it is ready for feature learning, and the time and frequency domain data is needed to be combined by concatenation:

$$\mathbf{Xdata} = [\mathbf{x}; \mathbf{X}] \quad (3.6)$$

Herein, the data in (3.1) and (3.6) is utilized for feature learning. Feature learning is performed by using a non-iterative algorithm to obtain the features for classifier training and testing evaluations. This non-iterative algorithm is based on the ELM methodology.

The ELM is to find a discriminant function to classify the input example:

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\beta \quad (3.7)$$

The weights are given by

$$\mathbf{w}_k = [w_{k1}, w_{k2}, \dots, w_{ki}, \dots, w_{kR}]^\top \quad (3.8)$$

where R denotes the size of the weights and k is the k -th filter. The weights herein are randomly initialized. A traditional method to generate the weights is to follow the Gaussian distribution. Sometimes orthogonalization is needed to enhance the performance. With the random weights, the local receptive fields can be utilized to filter the data. The filtering process in essence is a convolution operation. An example of convolution is shown in Figure 3.2. Considering that the input is 1 dimensional, narrow convolution is performed for the k -th filter:

$$y_{kt} = \sum_{i=1}^R w_{ki} x_{t+i-1} \quad (t = 1, \dots, J) \quad (3.9)$$

The size J of the output $\mathbf{y}_k \in \mathbb{R}^J$ is determined by the formula:

$$J = (T + 2P - R) + 1 \quad (3.10)$$

FIGURE 3.2: An illustration of 1-Dimensional time series input for local receptive fields convolution with shared weights

where P is the padding factor, usually chosen to be 0. Shared weights are used to reduce the volume of the convolutional outputs. After convolution, the data is passed through the activation function

$$\mathbf{h}_c = g(y) \quad (3.11)$$

where $g(\cdot)$ is the activation function for the latent representations y . The process is then followed by a pooling function:

$$\mathbf{h} = pool(\mathbf{h}_c) \quad (3.12)$$

where $pool(\cdot)$ is the pooling function, which could be the max pooling, average pooling or square/square-root pooling. The matrix form of $\mathbf{H}$ can be used for M samples:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_M) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \dots & h_S(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_M) & \dots & h_S(\mathbf{x}_M) \end{bmatrix} \quad (3.13)$$

where S is the number of neurons.

The Local Receptive Fields (LRF) with random weights are utilized as feature learning. Instead of using alternative techniques such as dictionary learning and deep learning, ELM-LRF has been demonstrated to be a quick and effective feature mapping method [117]. The deep learning algorithms [118] [119], e.g. Convolutional Neural Network (CNN) [120] [121] [122], do require large datasets as training samples, and their back propagation is time-consuming in the training procedure.

The previous studies [123] [124] [21] [125] have shown that the random weights feed-forward neural networks can give a closed-form solution for classification problems. Accordingly, this methodology [126] [127] [128] is utilized instead of the deep neural networks for quick calculation and potential real-time applications.

With random weights, a non-iterative solution can be derived to determine the output β in the neural network by using Moore-Penrose pseudo-inverse: $\beta = \mathbf{H}^\dagger \mathbf{Y}$. An alternative solution is as follows:

$$\begin{aligned} \beta &= (\mathbf{H}^\top \mathbf{H} + \frac{\mathbf{I}}{\lambda})^{-1} \mathbf{H}^\top \mathbf{Y} \quad \text{if } M \gg S \\ \beta &= \mathbf{H}^\top (\mathbf{H} \mathbf{H}^\top + \frac{\mathbf{I}}{\lambda})^{-1} \mathbf{Y} \quad \text{otherwise} \end{aligned} \quad (3.14)$$

Convolutional Random Vector Functional Link Network (CRVFL) [26] is also a powerful tool that can be used as feature learning when considering randomized neural networks. It is demonstrated to be effective in video recognition. In CRVFL, the hidden neuron output is related to the input. This is based on RVFL [21] [24]. It is to make the input directly linked to the output layers, it appends the original $\mathbf{H}$ in (3.13) by using the input, concatenating into a new one:

$$\mathbf{H}_R = [\mathbf{H} \mathbf{X}] \quad (3.15)$$

where the size of $\mathbf{H}_R$ is $S \times (S + M)$. The derivation of RVFL is introduced herein. The output weights is achieved by minimizing the following optimization problem:

$$\mathbf{E} = \frac{1}{2M} \sum_{i=1}^M (y^i - \beta_R \mathbf{h}_R(\mathbf{x}_i)^\top)^2 \quad (3.16)$$

where y^i is the label and the hidden representation is:

$$\mathbf{h}_R(\mathbf{x}_i) = [h_1(\mathbf{x}_i) \dots h_S(\mathbf{x}_i) \ x_{i1} \dots x_{iM}] \quad (3.17)$$

In general, β_R can be determined by using both iteration method such as conjugate gradient method, or the Moore-Penrose pseudo-inverse: $\beta_R = \mathbf{H}_R^\dagger \mathbf{Y}$. Another closed form solution of the output weights by using the l_2 norm regularized least

square method is as follows:

$$\begin{aligned}\beta_R &= (\mathbf{H}_R^\top \mathbf{H}_R + \frac{\mathbf{I}}{\lambda})^{-1} \mathbf{H}_R^\top \mathbf{Y} \quad \text{if } M \gg S \\ \beta_R &= \mathbf{H}_R^\top (\mathbf{H}_R \mathbf{H}_R^\top + \frac{\mathbf{I}}{\lambda})^{-1} \mathbf{Y} \quad \text{otherwise}\end{aligned}\tag{3.18}$$

Finally, the trained β can be used for the testing procedure, with the testing samples processed in the same manner as the training samples using (3.9), (3.11), and (3.12). After transforming the testing samples into $\mathbf{H}_{test}$, the prediction can be predicted as

$$\hat{\mathbf{O}} = \mathbf{H}_{test} \beta \tag{3.19}$$

To sum up, the environmental sensing raw-data input is directly fed into the local receptive fields for feature extraction. Besides, input to the feature learning framework can be the combination of time-domain and frequency-domain raw data. After feature extraction, the classifier is trained to obtain the weights by using the labels and the learned features through the analytical solution. With the output weights, occupancy level prediction could be achieved.

3.2 Experiments and Discussions

3.2.1 Data Collection

Four sensors were deployed at two locations in a laboratory in the School of Electrical and Electronic Engineering, Nanyang Technological University. The layout is illustrated in Figure 3.3. Four types of environmental parameters were recorded by the non-intrusive sensors. They were *CO₂ concentration*, *Relative Humidity (RH)*, *Temperature* and *Air Pressure*. The sampling frequency was 1 sample per minute, which was 0.0167Hz. With this sampling frequency in a typical working day, 1440 samples were gathered with dimensions of 4. The regular occupancy pattern is shown in Figure 3.1, and it can be observed that three plateaus existed for the working hours as students would work from morning to evening whereas the two valleys were lunch and supper hours, respectively. It is noted that a few of the graduate students would work until late evening. The details of the environmental sensors, *CL11* and *MHB-382SD*, were as follows: *CL11* sensors measured the

CO_2 concentration with a resolution of 1 ppm with a variance of $\pm 5\%$ of the true value; $CL11$ sensors also measured the temperature and relative humidity with a resolution of $0.05^\circ C$ and $0.1\% RH$, respectively, and the variances were $\pm 0.3^\circ K$ and $< 2.5\% RH$, respectively. The air pressure sensor had a variance of $\pm 2 hPa$ with a resolution of $0.1 hPa$. Altogether, 30 working days of data was collected for evaluations.

As the environmental parameters are indeed sensitive to the condition of the room, the constraint is also provided in details. The suggestion is that sensors should be installed in public zone and not close to certain people in the office or lab. Besides, the windows should be closed at all time and the HVAC should be kept on as these are important to the environmental parameters.

FIGURE 3.3: Laboratory layout with sensors employed. Cameras were used for ground truth collection.

3.2.2 Experimental Setup and Evaluation Criteria

The program was run in the server using Matlab to examine the data collected from the sensors. The initial 25 days of data was utilized for training, and the remaining 5 days for testing. In the training dataset, a 5-fold cross validation was performed to tune the parameters. Each time, 1 fold was chosen as the validation set, and the remaining 4 folds as the training set. Through the evaluation on the validation set, the optimal parameters were chosen for the following experiments.

Firstly, preprocessing was applied to the raw data in (3.1). Initially, the length of samples in one day was 1440, i.e. $T = 1440$ based on the sampling frequency. As samples were gathered from four sensors in two locations, they were concatenated as a vector. Then, preprocessing was conducted in a 15-minute time step without overlapping, i.e. $N = 15$. Guidelines on how to decide on the value of N would be discussed in the subsequent experiments. Then, the time domain signals were appropriate for the proposed feature learning method in an autonomous manner. For the purpose of comparison with state-of-the-art approaches that would be discussed in the following experiments, feature engineering method is needed to extract features based on these preprocessed signals. Considering time domain feature extraction, the preprocessed signals were extracted by utilizing (3.2) with a time delay of 2, i.e. $s = 2$. The frequency domain features were not extracted for the traditional methods as there were no guidelines on how to design the features. Moreover, no earlier works have examined the utilization of frequency domain features in these feature engineering methods. With respect to feature learning methods, the frequency domain raw data may contain some valuable information that the neural networks can learn. The ELM-LRF was used for such purpose. The implementation of ELM-LRF was based on the code from Github¹ [117].

From the data in ground truths, the largest number of occupants in the dataset is 28. While it is good to achieve the estimation of the actual number of occupants, here, the occupancy is divided into a few occupancy levels because of the intended application in energy optimization and control which usually does not need the exact number of occupants. The four occupancy levels are adopted and they are zero, low, medium and high. It is adequate for the control strategy design of

¹<https://github.com/ExtremeLearningMachines/ELM-LRF>

TABLE 3.1: The division of occupancy levels based on the number of people

Occupancy Level	Number of People
Zero	0
Low	1~6
Medium	7~14
High	≥ 15

HVAC systems to apply this coarse information for energy saving. The reason why occupancy is divided into these levels are justified as follows:

Note that in industry standards, the demand-driven HVAC control is based on the recognition of presence and absence, i.e. basic binary occupancy detection. Even with this simple binary occupancy detection, significant energy savings have been demonstrated, e.g. the experiment in [6]. It is expected that significant energy savings can be realized with more defined levels of occupancy, where different ranges of thermal loads can be identified. A higher level is related to a higher load, for which more energy is required to maintain comfort at a temperature set point, whereas lower levels would require less energy. However, it is not necessary for the HVAC control system to take into account the exact occupancy numbers. For instance, lower occupancies such as 2 or 5 occupants would both represent similar low occupancy level information to the control system, whereas higher occupancies of 15 or 20 would represent similar high occupancy level information. Thus, a trade-off is necessary. In this manner, we adopted the division of the occupancy labels into 4 levels of zero, low, medium and high. The partition between the occupancy levels is based on thresholds which are set as $\{0, 1, 7, 15\}$ empirically. Zero means 0 occupant, low means 1 to 6 occupants, medium signifies 7 to 14 occupants and high means 15 or more occupants. This definition of occupancy level depiction is depicted in Table 3.1.

As for metrics of performance, the standard ones of Normalized Root Mean Square Error (NRMSE) [53], Testing Accuracy (Acc) and Detection Rate (DR) are applied. These metrics are calculated as follows:

$$NRMSE = \frac{\sqrt{\frac{\|\hat{\mathbf{O}} - \mathbf{O}\|_2^2}{T}}}{\mathbf{O}_{max} - \mathbf{O}_{min}} \quad (3.20)$$

where $\hat{\mathbf{O}}$ denotes the predicted levels of the occupancy, $\|\cdot\|_2$ means the l_2 -norm and the number T signifies the number of the samples in a day. Testing Accuracy

is defined as

$$Acc = \frac{T - \sum_{t=1}^T \mathbf{1}(\hat{O}_t - O_t)}{T} \quad (3.21)$$

where $\mathbf{1}(\cdot)$ is the indicator function

$$\mathbf{1}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.22)$$

Detection Rate (DR) is the accuracy of two states, occupied or vacant. This two-state detection is useful for lighting control for energy saving purpose [129]. The DR is computed as

$$DR = \frac{T - \sum_{t=1}^T |\mathbf{1}(\hat{O}_t) - \mathbf{1}(O_t)|}{T} \quad (3.23)$$

3.2.3 Results and Discussions

To demonstrate the effectiveness of our proposed approach, several existing feature engineering and feature learning methods were compared in this section. Firstly for the feature engineering methods, the time domain features in (3.2) were used, and the feature extraction implementation was as expressed in Section 3.2.2. The extracted temporal features were fed into the classifier model training process. The classifier models were HMM [50], DT [53], LDA [54], RVFL [21] [24], Multilayer Perceptron (MLP)[130] and ELM [48], respectively. In the work by Yang *et al.* [53], DT was one of the best classifiers, and LDA in the work by Candanedo *et al.* [54] was likewise one of the best classifiers. As a result, they were chosen to be contrasted with our outcomes. Secondly, the other frameworks using feature learning methods are also compared. For example, the work that is based on deep neural networks called CDBLSTM [130], which is also proposed by us, is further compared herein. As far as we know, ours are the first attempt to extract features automatically. The classifier model based on deep learning is Multilayer Perceptron (MLP). Besides, the feature learning using CRVFL [26] is also validated. For the purpose of comparison, the raw data in time domain was utilized as the feature learning algorithm inputs. The outcomes are summarized in Table 3.2.

It is necessary to determine the hyperparameters in the preprocessing stage. One of the most important parameters to be resolved is the length of the window time

TABLE 3.2: Comparison of Occupancy level prediction performance on the testing set

Literature	Methods	Model	NRMSE ↓	Acc(%) ↑	DR (%)↑	Time(s)↓
Dong <i>et al.</i> [50]	Feature engineering	HMM	0.1912	71.46	93.13	0.8913
Yang <i>et al.</i> [53]	Feature engineering	DT	0.2509	66.67	90.21	3.4345
Candanedo <i>et al.</i> [54]	Feature engineering	LDA	0.2297	70.21	88.54	0.1894
Masood <i>et al.</i> [48]	Feature engineering	ELM	0.2322	72.31	92.38	0.0707
RVFL [21] [24]	Feature engineering	RVFL	0.2013	72.50	92.92	0.2464
CRVFL [26]	Feature learning	RVFL	0.1990	73.19	92.67	0.6594
Chen <i>et al.</i>	Feature learning	MLP	0.1169	76.04	95.42	≥ 100
Proposed(time domain)	Feature learning	ELM	0.2042	74.50	93.63	1.2499

FIGURE 3.4: NRMSE comparison under different window length values.

FIGURE 3.5: Accuracy comparison under different window length values.

frame, i.e. N . Empirically, N is set to 15 minutes because the intended application in air conditioning system has slow dynamics, and a 15-minute time step is a decent solution as discussed in the work proposed by Liao *et al.* [28]. An examination of the various values of N has additionally been conducted to observe the variances correspondingly. The results of various values of N are shown in Figures 3.4, 3.5 and 3.6 with every technique compared. Among these three cases of $N = 10, 15$ and 30, the performance with $N = 15$ minute was a compromise. Consequently, N was set to 15 in subsequent experiments.

To ensure practical applicability, we have also examined the performance in the presence of noise. We found that almost all the data driven methods cannot effectively handle the noise problem. Our proposed framework also cannot resolve this issue. However, a noise removal algorithm can be incorporated into our proposed framework. The global smooth function proposed by Jiang *et al.* [47] was adopted to smooth the signal. The experimental results are tabulated in Table 3.3. It is

FIGURE 3.6: Detection Rate comparison under different window length values.

TABLE 3.3: Occupancy level prediction performance comparison with or without smoothing

Noise Removal	Model	NRMSE↓	Acc(%)↑	DR(%) ↑
No Smoothing	time domain LRF	0.2042	74.50	93.63
Smoothing	time domain LRF	0.1987	75.46	93.90

evident that after signal filtering with the smoothing function, the performance was somewhat improved. The accuracy was improved by 1% because of noise removal, and the detection rate was better. The NRMSE of the smoothing result was smaller than that without the smoothing operation.

The numerical outcomes of the overall experiment are summarized in Table 3.2, and it is observed that the feature engineering methods had an average testing accuracy of 70%. Amongst the feature engineering methods, HMM could provide a reliable result with a low NRMSE of 0.1912 and a high DR of 93.13%. This

(a) HMM with feature engineering

(b) DT with feature engineering

(c) LDA with feature engineering

(d) ELM with feature engineering and selection

(e) ELM-LRF feature learning

FIGURE 3.7: Comparison of occupancy level prediction using time-domain frameworks on the testing set

(a) RVFL with feature engineering

(b) CRVFL with feature learning

(c) CDBLSTM feature learning

(d) ELM-LRF feature learning

FIGURE 3.8: Comparison of occupancy level prediction using time-domain frameworks on the testing set (continued)

outcome made sense as it is consistent with the remark on the model of HMM commented by Yang *et al.* [52]. In the work studied by Masood *et al.* [48], feature selection was utilized on the temporal features with an ELM classifier to yield acceptable outcomes. The performance of RVFL and CRVFL is satisfactory. The best performance based on the time domain data only is the CDBLSTM feature learning framework. The ELM-LRF feature learning framework using only the time domain raw data gave the satisfactory performance of 74.50%. Furthermore, its DR was likewise satisfactory at 93.63%. From this basic assessment, it is evident that feature learning frameworks outperformed the feature engineering methods, without the need to utilize empirically designed temporal features.

The visualization comparison of the estimation result is shown in Figure 3.7 and 3.8. Based on the estimation of the occupancy level, we could find that DT gave a high variance output, whereas HMM provided a stationary yield. The feature selection based on the ELM strategy may be overfitting. Despite the fact that HMM could track the regular occupancy pattern, which was very accurate at the time of arrival and time of leaving, it could not track the stochastic occupant behaviors regarding entrance and exit during the working hours. In any working zone, one needs to manage the unavoidable stochastic nature of entrance and exit for some individual issues such as going to the toilet and filling water bottles. We remark that feature learning frameworks could catch this dynamic pattern accurately.

Note that all the techniques could give a decent estimation of the time of arrival and this is important to the HVAC control systems. The time of leaving is likewise of significance too, but numerous methods cannot estimate the leaving time accurately. The outcomes of LDA and DT indeed showed a few fluctuations at midnight. After a careful investigation, we found that a CO_2 sensor was placed somewhat near a graduate student who regularly stayed late until 11 p.m. Since the sensors were deployed behind his cubicle and therefore the CO_2 concentration was easily affected by this subject at night. Thus, it is reasonable to infer that some of the classifiers revealed a high occupancy level at night. However, because of the features learned in the framework, we found that the proposed framework could still handle this unusual condition and provided acceptable outcomes at night. In fact, the proposed feature learning algorithm predicted the time of leaving precisely for about 3 testing days. For this phenomenon, HMM could also track the occupancy dynamic patterns accurately because HMM depends on the transfer likelihood of

the occupancy levels. In this manner, it was not sensitive to the abnormal change of state from the low level to high level at night.

In terms of computing time, the various approaches were compared and the results are summarized in Table 3.2. From this investigation, it is apparent that the proposed feature learning framework had the best performance yet did not incur excessive computing time. The ELM classifier was one of the fastest models among these techniques. The computing time of the work of Masood *et al.* [48] is slightly faster but its performance is slightly worse than that of RVFL [21] [24]. One reason is that there is extra input concatenating into the hidden layer in calculating the output weights for the work of RVFL implementation. The proposed feature learning framework benefited from this non-iterative process in classifier training. Although the execution of the feature learning framework was not the fastest, it provided a trade-off between performance and efficiency. Consequently, the proposed feature learning framework is reasonable for real time application. The computing machine used in the experiment was the Intel Xeon CPU 2.70 (GHz) with RAM 512 GB. The implementation was based on Matlab scripts.

3.2.4 Time and Frequency Domain Data

In this subsection, the proposed feature learning framework was examined with both time and frequency domain data to investigate the impact of frequency domain data. The numerical comparison is depicted in Table 3.4. From the numerical results, it is clear that the frequency domain data alone could not give a better outcome, but they could significantly enhance the performance if concatenated with the time domain data. With combined time and frequency domain data, the testing accuracy was as high as 77.27%. It had the lowest NRMSE at 0.1842, and the Detection Rate was the most noticeable at 95.63%.

The evaluated results of the time and frequency domain data based on the feature learning framework on the testing set are shown in Figure 3.9. As aforementioned, the frequency domain data alone would not improve the estimation performance which is shown in Figure 3.9(b). But in Figure 3.9(c), with the combined time and frequency domain data, the feature learning framework could yield more steady outcomes than those utilizing only the time domain data in Figure 3.9(a). At the same time, the estimation of the time of leaving was additionally somewhat

TABLE 3.4: Occupancy level prediction performance comparison between time and frequency domain based feature learning on the testing set. (Freq is short for frequency)

Raw Data	Methods	NRMSE↓	Acc(%)↑	DR(%)↑
Time domain	ELM [48]	0.2050	69.33	93.71
	RVFL [21] [24]	0.2146	69.79	92.71
	CRVFL [26]	0.1990	73.19	92.67
	Proposed+ELM	0.2042	74.50	93.63
Freq domain	ELM [48]	0.2309	67.19	92.75
	RVFL [21] [24]	0.2302	68.33	93.33
	CRVFL [26]	0.2118	69.81	94.90
	Proposed+ELM	0.2122	69.33	94.67
Time and freq domains	ELM [48]	0.2134	69.29	94.02
	RVFL [21] [24]	0.2200	70.83	93.75
	CRVFL [26]	0.1858	76.83	95.81
	Proposed+ELM	0.1842	77.27	95.63

(a) Time-domain data based feature learning on the testing sets

(b) Frequency-domain data based feature learning on the testing sets

(c) Time and frequency domain based feature learning on the testing sets

FIGURE 3.9: Comparison of occupancy level prediction using feature learning with local receptive fields with random weights

FIGURE 3.10: Occupancy level prediction of time and frequency domain data based feature learning on the validation sets

enhanced. The average performance of the validation sets shown in Figure 3.10 was very accurate in terms of the regular occupancy level. It could accurately capture the pattern of the working hours.

In addition, RVFL and CRVFL are also validated with inputs from frequency domain and both time and frequency domains. From the Table 3.4, we can see that the performance of CRVFL in all the settings is also quite satisfactory. It is shown that the accuracy of RVFL is better than ELM. This conclusion has been demonstrated in recent works [131] [132] [133] [134]. When the frequency domain data is combined with the time domain data, it can indeed improve the performance of occupancy level prediction. In essence, we can conclude that the randomized neural networks are powerful tool that can be used as feature learning for the occupancy problem. It is demonstrated that with the help of additional frequency domain data, the feature learning framework can enhance the occupancy prediction performance to a higher level than the traditional methods which are only based on time domain data.

3.3 Summary

To sum up, we have proposed a framework to improve the occupancy level prediction by using a non-iterative LRF feature learning method. The feature learning method is based on the raw data to learn the distinguishable patterns automatically without the need to define engineered features. Consequently, it can be deployed with minimal domain-expert knowledge, which is different from the feature engineering methods that require empirical expert knowledge in the specific field to define the features. Moreover, we also find that when the temporal data

is combined with the frequency domain raw data as inputs to the feature learning method, the prediction performance can be significantly improved. The feature learning framework can learn representations of the dynamic patterns of occupancy levels more precisely, capturing the time of arrival, time of leaving and the stochastic exit and entrance during the working periods. Real experiments have been conducted with environmental sensing. The outcomes have demonstrated that this feature learning framework could effectively achieve satisfactory occupancy level prediction in a non-iterative manner, and could be implemented in real time.

Chapter 4

Simple Motion-related Activity Recognition with Inertial Measurements Using Semi-supervised Deep Learning

Besides the identification of occupancy levels, the identifications of occupants' simple motion-related activities and their interactions with utility resources within an office are also interesting problems. A good understanding of occupants' behaviors can provide much information for some potential applications. Compared to occupancy levels, occupants' behaviors are much more individual, but they have significant effect on energy consumption and thermal comfort [14]. Occupants' activities will affect their metabolism rates which are correlated to their thermal comfort sensation. For example, occupants would feel hot after doing some exercises and would have a high metabolism rate, thus requiring a cool comfort level of the environment. But those less active may have a lower metabolism rate, so preferring to have a medium comfort level of the environment. This information on occupant behavior is especially useful when considering the control and management of small zones such as offices and meeting rooms in a building. Unfortunately for this problem, the environmental sensing alone is not sufficient to understand their behaviors.

Thus, other sensing modalities and techniques are needed to study the simple motion-related activity recognition problem. Inertial measurement units embedded in smartphones are non-intrusive to occupants, and can therefore be used to study occupants' simple physical motions.

In this chapter, the occupant's simple motion-related activity recognition is studied by using a semi-supervised learning method. Traditional methods to recognize simple activities are to use supervised learning methods based on statistics as low-level features [61]. However, as unlabeled data is ubiquitous, how to leverage them is of practical interest. Besides, whether high-level features can be extracted to further enhance the performance of the recognition system is also of significant interest.

4.1 Methodology

In this section, we present the problem formulation of simple motion-related activity recognition using deep learning methods. Temporal ensembling of CNN is proposed by Laine *et al.* for image classification [135]. We further explore this structure and modify it for the time series signal classification. All the training samples are first augmented to increase the sample size. Then simple statistics are extracted as low-level features for occupant activity recognition. A Deep Long Short-term Memory (DLSTM) network is then followed to learn the high-level features. Besides, a dropout approach is applied as a regularizer to increase the generalization of the deep learning architecture. On one hand, the supervised loss can be calculated based on the predictions and labels. On the other hand, the unsupervised loss is calculated based on a comparison between unlabeled predictions and the ensemble predictions from the past epochs. The last layer is the softmax dense layer to provide classification outcomes. In addition, the supervised and unsupervised losses are combined to determine the parameters of deep neural networks using the back propagation method. The architecture of the proposed methodology is as illustrated in Figure 4.1. We elaborate on the proposed algorithm and discuss each part of the flowchart in the following subsections.

FIGURE 4.1: Architecture of the proposed temporal ensemble of DLSTM for occupant simple motion-related activity recognition

4.1.1 Deep Learning for High-Level Features Extraction

Our deep learning methods are based on LSTM. The LSTM was first proposed by Hochreiter and Schmidhuber in 1997 [136]. The recurrent structure is designed to capture the local dependency of the signals. Because of some gates and the memory cell state inside LSTM, the information can be persistent for a while. Due to the advantages of deep recurrent neural networks, LSTM is widely used to enhance the performance of recognition systems for its representative property [137].

In our proposed approach, both labeled and unlabeled datasets are processed. The labeled data is denoted as $\mathcal{X}_l = \{\mathbf{x}_l | l = 1, \dots, L\}$, where L is the number of labeled data. The labels are denoted as $y_l \in \{1, \dots, C\}$, where C is the number of classes in total. For unlabeled data, it is denoted as $\mathcal{X}_u = \{\mathbf{x}_u | u = 1, \dots, U\}$, where U is the number of unlabeled data.

Our DLSTM network structure is as shown in Figure 4.2, and we formulate it as follows. We assume that the deep neural networks have K layers. Each layer is concatenated to obtain the deep architecture. The inputs of the recurrent networks

FIGURE 4.2: The DLSTM structure

FIGURE 4.3: The LSTM inner structure

contain both labeled and unlabeled data, denoted as $x_t \in \mathcal{X}$ at time step t , where $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$. And the outputs of the k -th layer of LSTM are denoted as h_t^k . The inner structure of LSTM is shown in Figure 4.3. Three gates, i.e. input gate, forget gate and output gate, are parts of LSTM to process the persistence memory of the past content. They are denoted as i_t^k , f_t^k and o_t^k , respectively. Their definitions are as follows:

$$\begin{aligned}
i_t^k &= \sigma(W_i^k[h_{t-1}^k, h_t^{k-1}] + b_i^k) \\
f_t^k &= \sigma(W_f^k[h_{t-1}^k, h_t^{k-1}] + b_f^k) \\
\tilde{C}_t^k &= \tanh(W_C^k[h_{t-1}^k, h_t^{k-1}] + b_C^k) \\
C_t^k &= f_t^k * C_{t-1}^k + i_t^k * \tilde{C}_t^k \\
o_t^k &= \sigma(W_o^k[h_{t-1}^k, h_t^{k-1}] + b_o^k) \\
h_t^k &= o_t^k * \tanh(C_t^k)
\end{aligned} \tag{4.1}$$

The current network output can be used as the next layer input, and the first inputs are defined as:

$$h_t^0 = x_t \tag{4.2}$$

4.1.2 Generalization based on Randomness

Three operations, i.e. Gaussian noise layer, data augmentation and dropout, are embedded in the deep architecture so as to make good use of unlabeled data to measure the unlabeled loss. They can be regarded as regularizers, introducing randomness inside the neural networks for evaluating the unsupervised loss.

Firstly, the Gaussian noise layer is added in the input layer. By adding the Gaussian noise to the inputs randomly, the neural networks gain some robustness to avoid overfitting. Thanks to the perturbation, the networks can also be regularized as stated in the previous work [138]. Besides, after introducing randomness, the outputs of the neural networks become stochastic variables, which can be applied to evaluate the difference between the predictions and the outcomes from unlabeled data to obtain a proper unsupervised loss.

Secondly, data augmentation also acts as a regularizer in a stochastic manner by introducing a random scaling ratio to rescale the magnitude of the signals. Furthermore, it can enlarge the number of the training sets so as to improve the performance of the neural networks. The data augmentation of time-series data is

to use window slicing or window warping as discussed in the previous work [139]. A sample can be sliced into many frames. The window slicing process can also overlap. This is a typical method to increase the number of training samples when dealing with the time series data [61].

Thirdly, the dropout can be regarded as an ensemble process [140]. The illustration of the dropout in the neural networks is shown in Figure 4.4(a) and Figure 4.4(b). Some neurons are neglected by random selection in a probabilistic manner, e.g. the shaded hidden nodes which are discarded. If it is run for many times, then the random dropout networks form an ensemble process. Although it is different from the subset operation in the bagging ensemble method, it produces many different outputs based on different neurons. As a result, dropout technique is added to increase the generalization of the neural networks with the probabilities denoted as p .

According to the study of Sajjadi *et al.* [141], the transformation and stability loss can be used to evaluate the unlabeled loss. Consequently, it is required to introduce randomness into the deep architecture. The three aforementioned operations can produce stochastic outputs. Besides, it is beneficial to add some noise to the inputs and to enlarge the size of the training numbers as these can enhance the generalization of the deep neural networks.

4.1.3 Losses and Temporal Ensemble

After the aforementioned operations, the output z_i of the networks for each sample becomes:

$$z_i = NN(x_i; \theta) \quad (4.3)$$

where $NN(.)$ is the neural network that tries to learn from the training samples. All the weights and bias of all LSTM layers in the neural networks are termed as θ , i.e.

$$\theta = [W_i, b_i, W_f, b_f, W_o, b_o] \quad (4.4)$$

The DLSTM plays an important role in learning the high-level features. The temporal ensemble process is based on the learned features at each time step. For the supervised learning loss, the standard cross-entropy is used to evaluate the

(a) Standard neural networks

(b) Dropout neural networks

FIGURE 4.4: The dropout illustration

difference between training labels and predicted ones. As a result, the supervised learning loss is defined as follows:

$$loss_l(\theta) = -\frac{1}{L} \sum_i (y_i \log(z_i) + (1 - y_i) \log(1 - z_i)) \quad (4.5)$$

As for the unsupervised learning loss, the predicted outputs can be compared with the previous ensemble outputs. It is the stochastic operations in the deep neural networks that make this loss work. And the square loss is one candidate to evaluate

the difference between them, i.e

$$loss_u(\theta) = \frac{1}{CU} \sum_i ||z_i - \tilde{z}_i||^2 \quad (4.6)$$

where $\tilde{z}_i$ is the temporal ensemble output from the previous time steps. It is the temporal accumulating ensemble of predictions of the last several steps. If the outputs are combined into the matrix form, i.e. $\tilde{\mathbf{z}}$, it can be updated with a normalization process following this operation:

$$\tilde{\mathbf{z}} = \frac{\mathbf{Z}_\tau}{1 - \alpha^\tau} \quad (4.7)$$

where $\mathbf{Z}_\tau$ is the latent variable at each epoch τ , which basically is the combination of previous outputs from the neural networks, and α is the momentum factor, acting as a rate to monitor the previous temporal ensembles. At first, $\mathbf{Z}_0$ is initialized as 0. Then it can be updated using the following formula by accumulating the previous predictions:

$$\mathbf{Z}_\tau = \alpha \mathbf{Z}_{\tau-1} + (1 - \alpha) \mathbf{z} \quad (4.8)$$

where $\mathbf{z}$ is the matrix form of z . Combining the supervised and unsupervised learning components, the resultant loss with a weight can be defined as:

$$loss(\theta) = loss_l(\theta) + w(\tau) * loss_u(\theta) \quad (4.9)$$

where $w(\tau)$ is the weight parameter that is dependent on the time epoch. It should be kept small at the early stage of training of the deep neural networks and then ramped up after the early training phase, e.g. after 80 training epochs. The loss is to be minimized to determine the suitable parameters θ . This can apply the back propagation as studied by Rumelhart *et al.* [142] with gradient descent to determine the weights in neural networks. But in recurrent neural networks, the case now is different and a phenomenon called gradient vanishing appears. Hence the method of back propagation through time (BPTT) is chosen. It can be used to determine the weights and bias in the recurrent neural networks. The Adam [143] is employed in the optimization process with gradient clipping to solve the gradient vanishing problem.

Finally, a softmax classifier is utilized based on the learned features to provide a probabilistic interpretation, i.e.

$$p(Y = j|z_i, W, b) = \frac{\exp^{W_j z_i + b}}{\sum_{c=1}^C \exp^{W_c z_i + b}} \quad (4.10)$$

where W and b denote the weight matrix and bias vector of the softmax layer, respectively, and $Y = j$ denotes the stochastic variable Y belonging to the j -th class. And the model selects the prediction with the maximal probability:

$$\hat{y}_i = \arg \max_j P(Y = j|z_i, W, b) \quad (4.11)$$

To sum up, the inertial sensor raw-data inputs are firstly extracted by using statistics as low-level features. Then, they are fed into the DLSTM neural networks to extract high-level features. With these features, supervised learning loss is calculated by comparing neural network outputs with labels of labeled samples. In addition, unsupervised learning loss is calculated by comparing the outputs of the neural network with the past ensemble predictions of unlabeled samples. Both the supervised and unsupervised losses are combined into a total loss. This loss is minimized by using BPTT to determine the weights in the deep neural networks. With the trained weights, these neural networks can predict the classes of the testing inertial sensing samples.

The contribution of this method is that the DLSTM with temporal ensembling is a new approach proposed by us that can effectively solve the semi-supervised learning problem for the simple motion-related occupant activity recognition. Firstly, semi-supervised problem is important because labeled data is too expensive to be collected in terms of motion sensors while unlabeled data is ubiquitous, and the proposed method is different from the supervised learning methods. Secondly, the proposed method combines with other existing methods in a meaningful way to extract high-level features automatically for semi-supervised learning problem and it is different from the traditional semi-supervised learning methods which are based on low-level features. Besides, the proposed method is different from the temporal ensembling of CNN proposed by Laine et al. [135] as their method was designed for the two-dimensional image signals but not directly suitable for the temporal motion sensor signals. Last but not least, the performance of the proposed method is demonstrated with real experiments.

4.2 Experiments and Discussions

The proposed method was evaluated using real experimental data and compared with state-of-the-art semi-supervised methods. The dataset and the deep architecture are first described in details, especially some hyperparameter settings, then the experimental results are reported and analyzed. Finally, the effects of some of these parameters on the performance of the system are discussed.

4.2.1 Dataset

The public dataset studied in this section was based on the data collected using inertial sensors of a smartphone, with a total of 30 subjects ranging from 19 to 48 years old [61]. The smartphone used was Galaxy S II, and the device was attached to the waist of subjects. The six activities of interest are *Sitting*, *Laying down*, *Walking*, *Walking upstairs*, *Walking downstairs* and *Standing*. The built-in sensor included accelerometer and gyroscope at a sampling rate of 50Hz. The numbers of training samples and testing samples were approximately 7000 and 3000, respectively.

For each time-series instance, it could be sampled using window framing with a certain percentage of overlapping. Empirically, the fix-width window could enrich the performance because this window could cover a full action period, e.g. walking cycle. Following the instructions of the previous work proposed by Anguita *et al.* [61], the same parameter settings were chosen. Specifically, the length of the sliding window was 2.56 seconds, and the overlapping rate was 50%.

Basic statistics were calculated for each instance to extract the low-level features, including both time and frequency domains features. A summary of the computing features can be seen in Table 2.2. Some normal statistics were extracted, including the maximum, minimum and mean of the amplitudes.

4.2.2 Experimental Setup

Three semi-supervised learning methods were implemented for comparison, namely Co-training [81], Self-training [81] and Label Propagation (LP) [82]. The toolbox

TABLE 4.1: Comparison between traditional semi-supervised learning methods and the proposed method

Methods	Labeled percentages										
	1%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
LP [82]	76.76	83.61	86.94	88.19	88.80	88.97	88.67	86.26	84.46	89.51	89.82
Self-training [81]	81.98	81.61	84.02	86.16	88.60	88.63	89.89	89.55	89.85	90.50	90.16
Co-training [81]	81.74	91.75	92.53	92.94	93.79	93.62	94.27	94.16	94.10	94.13	93.99
Proposed	87.67	94.74	95.99	96.16	96.67	96.88	96.71	97.04	97.11	97.11	97.21

used was based on Scikit learning library in Python. Co-training is a two-view method by processing different views for predictions in an iterative manner and the most confident one is used [81]. The feature was separated into two views and the Random Forest classifier was used with the number of trees being 500. Self-training [81] was used with the K-nearest-neighbor (KNN) Classifier, with the number of neighbors $k = 5$. Label Propagation [82] was used with a KNN kernel where $k = 6$ and the clapping factor $\alpha = 0.8$.

The proposed method is called the temporal ensemble of DLSTM (TEDLSTM) and it is evaluated against traditional semi-supervised learning methods. The deep architecture was implemented based on Lasagne [144] using Python. The number of the layers was set to be 2 empirically. The probabilities of the two dropout layers were $p = (0.2, 0.2)$. The other parameters followed the settings designed by Laine *et al.* [135]. Every layer applied the nonlinear leaky rectifier linear unit (ReLU) [145] with a value of 0.1. And the initialization followed the method proposed by He *et al.* [146]. Furthermore, after the input layer, Gaussian noise layer was added with a standard variance of 0.2. Besides, the maximal learning rate of the Adam optimization was set to be 0.003 with the Adam momentum parameters being (0.9, 0.999). The maximal weight of the unsupervised loss was set to be 50. The total epoch was 150 with the early stopping criteria.

4.2.3 Semi-supervised Learning Comparison

The performance of the semi-supervised learning algorithms is reported based on the increasing percentage of the training samples. The following settings are used: 1%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100% of the training samples. More details are summarized in Table 4.1. The results are also shown in Figure 4.5. From these results, some key phenomena can be observed and illustrated as follows:

FIGURE 4.5: Comparison with various semi-supervised learning methods

- From Figure 4.5, it is clear that the proposed method outperformed all the other methods for all percentages of the training samples. As the percentages of the training sample were increased, all the semi-supervised learning methods achieved better outcomes.
- For the scenario with a small percentage of labeled data, e.g. with only 1% the training samples, the advantage of our proposed method was apparent, and it achieved about 11%, 6% and 6% improvements in accuracies when compared with the Co-training, Self-training and Label Propagation approaches, respectively.
- When all the training samples were fed into the proposed method, it achieved the best result of 97.21%. This performance was also the best result among all the compared methods. From this result, it is clear that our proposed method was better than the other semi-supervised methods.

The confusion matrix of the proposed method with all the training samples is shown in Figure 4.6. From a comparison between predicted labels and true labels, it is clear that the proposed method could easily distinguish the Laying down activity among all other activities. It could be seen that the Walking category was

FIGURE 4.6: Confusion matrix of our proposed method

recognized well, even though it was somewhat similar to the Walking upstairs and the Walking downstairs except for the vertical difference in the acceleration data. It is known that if one puts the sensor at the waist, it is somewhat difficult to tell the difference between the Sitting and Standing because the most significant physical movement relies on the movement of the knee. And this was also found to be the case in the work of Cleland *et al.* [147], which indicated that the lower torso is the best on-body placement for human activity recognition using inertial measurement units. Nonetheless, the proposed method could still achieve a satisfactory result in distinguishing the Sitting and Standing.

4.2.4 Empirical Parameters

It is useful to examine the effects of some hyperparameters in the proposed architecture. The most important parameters were chosen to study their effects on the performance of the system, and these parameters were the number of layers and the probability of the dropout.

The number of layers of 1 to 5 was considered. The performance with different number of layers is shown in Figure 4.7. It could be readily seen that when the number of layers was 2, it achieved the best performance. Hence, it was selected as the default number of layers in all other experiments. In fact, as the number of layers was increased, the neural networks could easily be overfitting. Further, with a larger number of layers, more computing time would be required in the training process without yielding better performance.

FIGURE 4.7: Performance of deep neural networks with different number of layers

As for the probability of dropouts, an experiment was designed to observe its effect on the performance of the system. The results of different combinations of two dropout layers are shown in Figure 4.8. Specifically, we denoted $p = 0.2$ as low level, $p = 0.5$ as medium level and $p = 0.8$ as high level, respectively. From the performance comparison with different probabilities of dropouts, we noted that the difference in performance was not large. The best performance was 97.21% with the dropout probabilities of (0.2, 0.2). The value of 0.2 was a typical value in training the neural network, whereas 0.8 was considered large with too much noise inside. As a result, the performance with $p = (0.8, 0.8)$ was only 93.85%.

The computing time required was also analyzed. The testing time for around 3000 testing samples was 2.118s. That is to say, the testing phase could be done in

FIGURE 4.8: Deep neural networks performance with different probability of the dropout layers

real time, with a time cost of $0.7ms$ per instance. Although the training time is a little bit longer than other semi-supervised learning methods, it is still effective and manageable since it is trained only once. It is time-consuming to determine the weights of deep neural networks. But the testing process is some simple multiplications and additions using the trained weights. As a result, considering the classification performance, the deep neural networks are still a satisfactory method. The computer used was with the duo core CPUs of Intel Xeon at $2.70GHz$ and a GPU of NVIDIA K40.

4.3 Summary

In summary, we have proposed a deep semi-supervised method for simple motion-related activity recognition. Specifically, the type of activities considered are *Walking*, *Walking upstairs*, *Walking downstairs*, *Sitting*, *Standing* and *Laying down*. We performed the relevant activity recognition by processing the information gathered

by inertial measurement units embedded in smartphones. The proposed semi-supervised learning method is combined with DLSTM to extract high-level features. Besides feature extraction, a temporal ensemble method is investigated with randomness inside to enhance the generalization of the neural networks. The outputs of the neural networks are compared and evaluated against the past ensemble predictions so as to calculate the unsupervised learning loss. A joint loss of supervised learning and unsupervised learning is then used to determine the weights of the recurrent neural networks via back propagation through time. After training with both labeled and unlabeled data, even when the percentage of labeled data is small, the proposed system performed well. The experimental results showed that the proposed method outperformed all the other three semi-supervised learning methods.

Chapter 5

Simple Motion-related Activity Recognition with Inertial Measurements Using Dictionary Learning

In this chapter, we continue to explore the problem of simple motion-related activity recognition but now with dictionary learning. Same as the focus in Chapter 4, simple motion-related actions of walking, running and other activities based on inertial sensing are studied. Since the occupants' behaviors are closely related to their daily lives in buildings, their recognition problem could be addressed through the use of smartphones as a mobile computing platform. We demonstrate that the dictionary learning could improve the performance of simple motion-related activity recognition based on our dataset collected by using smartphones.

5.1 Methodology

In this section, the smartphone-based occupant simple motion-related activity recognition problem is formulated by following the work on Locality-constrained Linear Coding (LLC) [148]. The dictionary learning method, e.g. LLC, is based on clustering as a basis to encode the low level features into high level ones. This

clustering process is usually based on the K-means algorithm [149]. And the encoding process can have various methods to obtain the codings. For example, vector quantization and sparse coding are two popular dictionary learning methods to obtain the codings for many classification tasks. But LLC is different from these two methods. One of the key properties of LLC is that it has better reconstruction than vector quantization. Moreover, it also has locally smooth sparsity as compared to sparse coding. Besides, it gives an analytical solution which can be solved in a non-iterative manner. The dictionary learning methods are mainly based on the concept of bag-of-features, and we aim to apply them for the occupant simple behavior recognition. The architecture of the smartphone-based simple motion-related activity recognition using the LLC dictionary learning is shown in Figure 5.1.

FIGURE 5.1: Architecture of LLC-based framework of smartphone-based occupant simple motion-related activity recognition

Given a time series of sensors measurements $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_N]$, where $\mathbf{f}_i \in \mathbb{R}^l$ denotes the i -th frame with l dimensional, feature extraction could be performed to transform the raw data into a feature denoted as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$, where $\mathbf{x}_i \in \mathbb{R}^D$ denotes the i -th sample's features with dimension D . The traditional feature extraction methods can be utilized here. In order to represent the features in a more reliable manner, the dictionary learning method can be applied to encode the low-level features into better codes. By clustering, a dictionary $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_M]$ can be obtained, where $\mathbf{b}_i \in \mathbb{R}^D$ denotes the i -th codebook with dimensional D . The original low-level features can be further factorized based on the basis vectors:

$$\mathbf{x}_i \approx \mathbf{B}\mathbf{c}_i \quad (5.1)$$

where $\mathbf{c}_i \in \mathbf{R}^M$ is the i -th code, and M is the number of features in the dictionary. Because the samples from the same class are probably neighbors in the feature

space, the original feature can be encoded by imposing the locality constraint and the linear coding optimization is as follows:

$$\begin{aligned} \arg \min_{\mathbf{c}} \sum_{i=1}^N \|\mathbf{x}_i - \mathbf{B}\mathbf{c}_i\|_2^2 + \lambda \|\mathbf{d}_i \odot \mathbf{c}_i\|_2^2 \\ s.t. \mathbf{1}^T \mathbf{c}_i = 1, \forall i \end{aligned} \quad (5.2)$$

where $\odot$ signifies element-wise multiplication, λ is the regularization parameter, and

$$\mathbf{d}_i = [\exp(\frac{\text{dist}(\mathbf{x}_i, \mathbf{b}_1)}{\sigma}), \dots, \exp(\frac{\text{dist}(\mathbf{x}_i, \mathbf{b}_M)}{\sigma})]^T \quad (5.3)$$

where $\text{dist}(\mathbf{x}_i, \mathbf{b}_i)$ denotes the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{b}_i$, and σ is the decaying parameter of the exponents. This optimization problem has an analytical solution:

$$\boldsymbol{\alpha}_i = (\mathbf{Cov}_i + \lambda \text{diag}(\mathbf{d})) \setminus \mathbf{1} \quad (5.4)$$

$$\mathbf{c}_i = \boldsymbol{\alpha}_i / \mathbf{1}^T \boldsymbol{\alpha}_i \quad (5.5)$$

where $\mathbf{Cov}_i = (\mathbf{x}_i \mathbf{1}^T - \mathbf{B})^T (\mathbf{x}_i \mathbf{1}^T - \mathbf{B})$ denotes the covariance of the features. This analytical solution can be derived by using Lagrange Multiplier and Fisher Linear Discriminant Analysis (LDA) [150]. The analytical solution can be computed quickly when compared with other sparse coding and representation algorithms which use l_1 term as a regularization. After solving this optimization problem, the corresponding codes can represent the samples as high-level features. Note that the dictionary learning here does not require labels in the encoding process, as it is purely an unsupervised learning method in the feature extraction process.

Whereas the LLC-based feature learning framework for occupant simple behavior recognition has an analytical solution, an even faster encoding process can be achieved by using the approximated LLC [148] instead of the previous one. Considering that the classification algorithm could be computed using smartphones, a fast algorithm is favored. The approximated LLC is to approximate the regularization term in (5.2) by using S ($S \ll M$) nearest neighbors basis vectors. In this manner, the computation complexity could be reduced from $\mathcal{O}(M^2)$ to $\mathcal{O}(M + S^2)$. Hence, this straightforward form of the approximated LLC is applied in the proposed architecture to implement fast computations. The optimization problem is

now simplified as follows:

$$\begin{aligned} \arg \min_{\tilde{\mathbf{c}}} \sum_{i=1}^N \|\mathbf{x}_i - \tilde{\mathbf{c}}_i \mathbf{B}\|_2^2 \\ s.t. \mathbf{1}^T \tilde{\mathbf{c}}_i = 1, \quad \forall i \end{aligned} \quad (5.6)$$

If the choice of S is larger than D , then this problem is ill-posed. In order to make sure that every choice of the parameter S has an optimal solution, a regularization term $\lambda \|(\mathbf{x}_i \mathbf{1}^T - \mathbf{B})^T\|_F^2$ can be added in (5.6).

After obtaining the codes for each sample, it can be combined with their corresponding labels, $y_i \in \{1, 2, \dots, K\}$, into a set $\{\mathbf{c}_i, y_i\}$ for the following classifiers training, where the K denotes the number of classes.

We shall elaborate on the classifier training by introducing the Kernel Ridge Regression (KRR) [151] [152] herein. Based on the formulation of KRR in the survey [23], we follow their works and show some details. A typical classifier model is to solve the following minimization problem:

$$\arg \min_{\mathbf{w}} \sum_{i=1}^N (\mathbf{w}^T \mathbf{x}_i - \mathbf{y}_i)^2 + C \|\mathbf{w}\|_2^2 \quad (5.7)$$

The solution to this problem can be closed-form:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X} + C\mathbf{I})^{-1} \mathbf{X}^T \mathbf{Y} \quad (5.8)$$

where C is the constant. This is for linear features. If the features are not linearly separable, then a kernel method should be applied. That is, the KRR incorporates the kernel to transform the nonlinearly separable features into another feature space that could be separated linearly. The feature space $\phi(\mathbf{x}_i)$ can be used for the linear combination of the above solution: $\mathbf{w} = \sum_i \beta_i \phi(\mathbf{x}_i)$, then the problem above can be reformulated as:

$$\arg \min_{\beta} (\mathbf{Y} - \mathbf{K}\beta)^2 + C\beta^T \mathbf{K}\beta \quad (5.9)$$

The solution to this problem is also obtained in closed-form:

$$\beta = (\mathbf{K} + C\mathbf{I})^{-1} \mathbf{Y} \quad (5.10)$$

The kernel matrix $\mathbf{K}$ is defined as: $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$. After the LLC feature learning phase, we obtain the learned feature $\mathbf{c}_i$ and can feed it into (5.7) for the classifier training. Then we can trained the data to get the solution as models for the testing phase.

We list the equations of the kernel ELM [128]. The ELM is consistent with the ridge regression theory [153] so we use the same notations for kernel ELM. The solution is as follows:

$$\beta = (\mathbf{K} + C\mathbf{I})^{-1}\mathbf{Y} \quad (5.11)$$

5.2 Experimental Setup

In this section, our inertial dataset based on smartphone sensing platform is firstly described. Then typical feature extraction method is introduced to obtain the statistical features in both time and frequency domains. Furthermore, the LLC-based framework using the approximated solutions in implementation is introduced.

5.2.1 Dataset Collection

The three-dimensional accelerometer and gyroscope data for this study was collected with an Android application for a Google smartphone, NEXUS 4. Five subjects with ages ranging from 22 to 31 years old were involved in the experiment and they performed five activities of *static*, *walking*, *running*, *walking upstairs* and *downstairs*. The sampling frequency was $20Hz$. The subjects carried the smartphones in their trousers' front pockets and they performed the activities in the same manner as they did in their normal daily lives. The dataset collection procedure was approved by the Institute Review Board (IRB) of the university because the study involved human subjects. Every subject signed a consent form of participation for this data acquisition experiment and received a cash reward. A few measurements of the activities in terms of raw signals are illustrated in Figure 5.2. From the raw data, it is difficult to identify its class, thereby requiring feature extraction. Based on the raw signals, features could be extracted to represent the samples in a distinguishable manner.

FIGURE 5.2: Raw acceleration signals of a walking sample

5.2.2 Feature Extraction and Clustering

After data acquisition, a sliding window operation was performed to segment the raw signals. The sliding window was of 5-second duration with an overlapping rate of 50%. As a result, each segment contained 100 sampling time steps with three-dimensional accelerometer and gyroscope data. The time and frequency domain features as depicted in Table 5.1 were extracted for each segment. The recent work proposed by Anguita *et al.* [61] showed that these standard features were useful for statistical pattern recognition, so these features were first extracted.

All features were then clustered into a codebook for the LLC encoding using K-means algorithms [149]. In fact, the dictionary should be optimized to determine the optimal solution, but in our implementation, K-means clustering indicated satisfactory results. Consequently, the optimal dictionary was not searched in this study but instead a computationally lightweight implementation was adopted.

TABLE 5.1: Time and frequency domain feature extraction

Domain	Features
Time	Absolute Mean
	Median Absolute Deviation
	Maximum
	Minimum
	Signal Magnitude Range
	Interquartile Range
	Power
Frequency	Maximum
	Mean
	Kurtosis
	Skewness Power

5.3 Experimental Results

In this section, experimental results are presented for the classification performance of four classic classifiers. They are SVM [154], KNN [149], Kernel-ELM [128] and Sparse Representation (SR) Classifier [59]. In addition, KRR [152] [23] is also compared. Then an experiment by combining gyroscope data with accelerometer data is presented. Comparative results are then presented to show that the gyroscope data is able to enhance the performance of the recognition system. Last but not least, the outcomes of using LLC as feature learning are demonstrated to be effective.

5.3.1 Overall Evaluation

The entire dataset comprised 90 samples per class, i.e. a total of 450 samples for the 5 classes. Through feature extraction, a classifier using existing machine learning algorithms could be trained. Among the so many supervised learning algorithms, four typical ones as mentioned before were compared. For the first experiment, the acceleration features were extracted, which is shown in Table 5.1. Because the dimensions of the samples were not large, approximately 40, the classification algorithm could run in real time. The classification outcomes of these classifiers are depicted in Figure 5.3. Except for the sparse representation classifier, the other approaches could achieve approximately above 80% accuracy. As indicated by Morales *et al.* [65], the classification outcomes could be enhanced with the

FIGURE 5.3: Classification performance on the statistical features of acceleration data. The x-axis indicates the training number in each class.

gyroscope data. Therefore, the usefulness of gyroscope signals was further assessed in the following subsection.

FIGURE 5.4: Classification performance on the statistical features of accelerometer and gyroscope data

FIGURE 5.5: Comparison of classification performance with and without the gyroscope data

FIGURE 5.6: Comparison of classification performance with and without the gyroscope data (Cont'd)

5.3.2 Accelerometer with and without Gyroscope

The gyroscope features were concatenated to the accelerometer feature vectors. The dimensions of these feature vectors were therefore doubled to 80. In Figure 5.4, we report the classification results of the four previously mentioned methods after adding the gyroscope features. This time, the performance of every approach was improved, with some of the methods yielding about 90% accuracy. The improvement is clearly illustrated in Figure 5.5, where a comparison of performance with and without the gyroscope data is depicted. With the extra gyroscope information, the recognition accuracy was improved by 5%. But in the next subsection, it is shown that the outcomes would be further enhanced if the LLC feature learning is used. All the classifiers could give satisfactory results and is demonstrated to be improved to some extent when considering the gyroscope data. Both the Kernel ELM and KRR could give satisfactory results, which demonstrated that the functionality of the gyroscope. Their results are in the same level in that they are consistent as aforementioned. The implementation of KRR followed the work [152] [23], which can be accessed from the website¹ as well as in the appendices. The codes are based on the open source community.

FIGURE 5.7: Comparison of classification performance of acceleration data between statistical features and LLC learning

¹<https://github.com/Alexzhuqch001/randomnn>

FIGURE 5.8: Comparison of classification performance of acceleration data between statistical features and LLC learning (Cont'd)

FIGURE 5.9: Comparison of classification performance of accelerometer and gyroscope data between statistical features and LLC learning

FIGURE 5.10: Comparison of classification performance of accelerometer and gyroscope data between statistical features and LLC learning (Cont'd)

5.3.3 Statistical Features Versus LLC Feature Learning

Based on the statistical features, the LLC codings could be obtained through the dictionary basis. The approximated form of the LLC encoding scheme was adopted. The results with only the acceleration data are presented in Figure 5.7 and 5.8. When the number of training samples in each class was increased, especially when the training number was 30 or higher, the performance of the LLC-based framework outperformed the one utilizing statistical features. With the extra gyroscope data, the results are presented in Figure 5.9 and 5.10. The improvement by using the LLC-based framework was approximately 5%, yielding classification performance accuracy of 90%.

Two more experiments for the entire dataset were conducted. In the first one, the confusion matrix is shown in Table 5.2. In this experiment on the overall performance of LLC, 80% of the whole dataset was used as training data and the remainder as testing data. It was run for 10 times to obtain the average outcomes. This general test using the LLC learning for the combined accelerometer and gyroscope features could achieve an average classification accuracy of 94.5%. Furthermore, the leave-one-subject-out cross validation strategy was conducted for these four classifiers using the LLC-based framework with combined accelerometer

TABLE 5.2: Confusion matrix for LLC-based accelerometer and gyroscope features

	Predicted Activity					Recall
	Down	Running	Static	Up	Walking	
Down	183	1	0	4	12	91.50%
Running	1	199	0	0	0	99.50%
Static	0	0	200	0	0	100%
Up	5	1	0	183	11	91.50%
Walking	8	0	1	11	180	90.00%
Precision	92.89%	99.00%	99.50%	92.42%	88.67%	

TABLE 5.3: Leave-One-Subject-Out average classification performance

SVM [154]	SR[59]	Kernel ELM [128]	KRR [23, 152]	KNN [149]
74.66%	76.22%	84.67%	84.67%	78.89%

and gyroscope features. The data from 4 subjects was selected as training data and the remaining one as testing data. The average results are depicted in Table 5.3. Clearly, the experimental results demonstrated that the LLC learning method worked well.

5.3.4 Limitation

The IMU is easily affected by the users, and the on-body placement of the users is fixed in their pockets or bags during the experiment. This is the limitation of the proposed framework. There are other on-body placements, for example, hand-held and fixed in arm for sports and so on. The difference among them can be solved by transforming their coordinates to the principal component axis as studied in another work [155]. To apply this technique, the result can be enhanced in a robust way. Although it did not consider the on-body variations, it can still recognize the movement of the simple activities of the users without too much restriction on their daily behaviors in certain placement.

5.4 Summary

In this chapter, LLC is applied as feature learning to realize smartphone-based simple motion-related activity recognition. The performance of the LLC learning in solving this problem is satisfactory. To demonstrate the effectiveness of our simple motion-related activity recognition, a dataset was collected by using the embedded accelerometer and gyroscope of smartphones. The features are extracted in a conventional manner but the LLC encoding scheme is conducted for high-level feature learning. The experimental outcomes demonstrated that the gyroscope signal can somewhat enhance the system's accuracy and the LLC-based framework could indeed improve the recognition performance.

Chapter 6

Energy-related Sound Event Recognition Using Dictionary Learning

In this chapter, occupant behavior recognition in terms of identifying the occupants' interactions with electrical appliances or utility resources is studied based on the sound signals recorded by built-in microphones of smartphones. We call this the energy-related sound event recognition problem. To realize smart building applications, more information about the occupants and their interactions with appliances should be mined to enrich the understanding of the occupants' activities as well as the indoor environment [14]. Unfortunately, IMU measurements alone cannot achieve the understanding of the occupant's interactive behaviors with electrical appliances. But the built-in microphones can be used to monitor these sound events as studied in the previous work [15]. Our approach to this energy-related sound event recognition problem is to use the same dictionary learning method studied in Chapter 5 but processed in a different manner.

6.1 Methodology

The energy-related sound event recognition framework is proposed by using the locality-constrained linear coding (LLC) method to improve the performance of the recognition system. Traditional methods to conduct sound event recognition

FIGURE 6.1: The LLC-based framework using unlabeled acoustic data for energy-related sound event recognition. The blue, red and green arrows denote the labeled training, testing and unlabeled data processing procedure, respectively.

are to apply some statistics as the low-level features. For example, the pitch and duration of the audios can be applied to perform simple sound event classification. Another popular method is to use the Mel Frequency Cepstral Coefficients (MFCCs) features that are originally designed for speech recognition [15]. In this chapter, LLC is applied to extract the high-level features for energy-related sound event recognition. In this application, labeled data may be difficult to obtain, while unlabeled data is pervasively available. It is therefore of great interest to explore how unlabeled data could be leveraged together with labeled data from smartphones to explore the sound event recognition problem using the LLC dictionary learning method.

The proposed architecture of our energy-related sound event recognition using smartphones is shown in Figure 6.1. The proposed framework includes the data collection, feature extraction and classifier training processes.

On data collection, there are mainly two datasets on energy-related sound events. The first one is the labeled dataset that is collected by using the built-in microphones of smartphones. After the recording process, the sound event sample is labeled by the users. Another dataset is the unlabeled one, which is from web-collected and user-recorded sound events but without labels. The two datasets can

be integrated into the LLC dictionary learning. The raw signals of some typical energy-related sound events are illustrated in Figure 6.2.

FIGURE 6.2: The raw acoustic signals of the 8 energy-related sound events

As for feature extraction, LLC is used to obtain the high-level features. After recording the raw acoustic data, the audio files were segmented into window samples with each of 25ms duration. For every window instance, the typical features of MFCCs were extracted based on the open toolbox [156], as these features are shown to be useful in audio activity recognition [157]. Figure 6.3 visualizes the MFCCs features. These features are the so-call feature-engineering for classification. Both labeled data and unlabeled data were utilized to extract the high-level features. They were clustered, and the centroids of the clusters became the dictionary for the LLC encoding. That is, the K-means clustering algorithm is utilized here to obtain the dictionary $\mathbf{B}$ from the entire dataset. This dictionary's centroids appeared to be satisfactory for our application, so we solved (5.2) directly by utilizing analytical solutions (5.4) and (5.5) without searching for the optimized dictionary. With the centroids of the clusters as the dictionary, we could encode every labeled feature into the LLC coefficients.

FIGURE 6.3: The visualization of the original signals using FFT, spectrum and MFCCs features of the audio file when occupants are working on personal computer

As far as classifier training is concerned, some typical classifiers could be trained to solve the energy-related sound event recognition problem. Combined with the labels, the high-level features could be fed into the traditional supervised learning algorithms. For example, in our experiment we applied SVM [154], ELM [128] and RVFL [21] [24].

In the following sections, our proposed framework is evaluated based on real experimental data.

6.2 Experimental Setup

6.2.1 Labeled Dataset

An application was designed on an Android smartphone for occupants to record and annotate the audio files. The users utilized a Samsung Galaxy S4 smartphone to record what they were doing by placing it near the sound source in a relative quiet environment. The sampling rate of the embedded microphone was about

TABLE 6.1: Index of occupant activity for energy-related sound event recognition

Index	Activity
1	Using coffee machine
2	Using electric fan
3	Using kettle
4	Using hair dryer
5	Using water tap
6	Flushing toilet
7	Printing papers
8	Working on personal computer

44.1kHz and the smartphone recorded each audio file of approximately 10 seconds duration. The format of the files was WAV. These files were transferred to the server. Annotations were made in a simple manner by asking the users to type the activities they performed into the smartphone. The 8 occupant behaviors we studied in this chapter are *Using coffee machine*, *Using electric fan*, *Using kettle*, *Using hair dryer*, *Using water tap*, *Flushing Toilet*, *Printing papers* and *Working on personal computer*. These are some common interactive activities between the occupants and utility appliances in the building. An index is used to denote each studied activity for convenience and it is as tabulated in Table 6.1.

6.2.2 Unlabeled Dataset

- Web-collected: Numerous but related audio files were downloaded from the website called Freesound¹. The files are ready-to-use from the Internet. The number of the downloaded files is 80. The duration of each file is approximately 30 seconds on average.
- User-recorded: This unlabeled dataset was recorded using the same smartphone but without any annotations. The majority of the unlabeled data is associated with the above mentioned 8 activities, with some extraneous sound events, such as *Having a conversation*, *Using microwave* and *Using vacuum cleaner*.

¹www.freesound.org

FIGURE 6.4: Comparison between LLC-based and Feature engineering-based method using SVM supervised learning

FIGURE 6.5: Comparison between LLC-based and Feature engineering-based method using ELM supervised learning

6.3 Experimental Results

6.3.1 Comparison between Proposed and Feature-engineering Methods

The comparison of classification performance between the LLC coefficients and feature-engineering (MFCCs) was evaluated. The performance of the recognition system using the SVM classifier is shown in Figure 6.4. In Figure 6.5, the performance of the recognition system using the ELM algorithm is visualized. We found that in both cases when the training number of each category was increased, the classification performance was improved for both the LLC-based and feature-engineering based approaches. We note that the LLC-based technique was significantly superior to the feature engineering approach. This showed that our proposed LLC-based framework was indeed suitable for audio recognition applications. Moreover, the LLC-based approach could deal with the case of few annotations. Indeed, even with a few annotations in each category, for example when the training number was 40, the LLC-based approach could achieve accuracies of 86% and 72% with the SVM and ELM algorithms, respectively, whereas the feature-engineering based approach yielded accuracies of 84% and 60%, respectively. Additionally, the classification performance is tabulated in Table 6.2 when all labeled data was used. The F_1 -score as the measurement metric is used herein [60]. We note that slight improvement was achieved when utilizing the LLC-based method as compared to the feature-engineering method. The LLC-based method could give an F_1 -score of as high as 92%. In addition, the performance of RVFL is satisfactory. When considering the LLC features, the results are improved to some extent. With the performance of these three classifiers, the LLC is demonstrated to be effective in energy-related sound event classification. Apparently, the LLC-based method had its advantages against the feature engineering method, especially in the situation when only a few annotations are available.

6.3.2 Performance on Web-collected Data

There is an abundance of audio files that can be downloaded from the website. Although the files in Freesound were annotated, only the samples of the acoustic

TABLE 6.2: Comparison between LLC-based and feature-engineering based methods using supervised learning algorithms on all labeled data

F₁-Score(%)	Feature Engineering	User-recorded	web-collected
SVM [154]	91.88	92.17	92.22
ELM [128]	80.98	90.14	89.61
RVFL [21] [24]	86.27	90.61	89.89

TABLE 6.3: Overall confusion matrix using web-collected dataset for dictionary learning

Activity Index	Predicted Activity								Recall
	1	2	3	4	5	6	7	8	
1	1479	86	12	4	19	46	31	3	88.04%
2	76	1495	0	0	21	79	9	0	88.99%
3	1	0	1556	38	11	72	0	2	92.62%
4	0	0	9	1633	2	36	0	0	97.20%
5	35	33	19	4	1533	20	28	8	91.25%
6	60	90	48	31	6	1429	13	3	85.06%
7	88	20	2	0	15	16	1539	0	91.61%
8	0	0	2	0	0	0	0	1678	99.88%
Precision	85.05%	86.72%	94.42%	95.50%	95.40%	84.16%	95.00%	99.06%	
F_1 -score	86.52%	87.84%	93.51%	96.34%	93.28%	84.16%	93.27%	99.47%	

data were used and labels were not used here. The LLC-based framework could leverage this unlabeled data for energy-related activity recognition. The overall performance of the classification is shown in terms of confusion matrix in Table 6.3. The index corresponds to the denotation given in Table 6.1. From the experimental results, it is clear that the proposed framework could give a satisfactory F_1 -score.

6.3.3 Performance on User-recorded Data

The effect of extraneous sound events is investigated on the unlabeled dataset. The users recorded their daily activities using smartphones. The dictionary was trained by adding the extraneous data. Unlabeled data was recorded by the users, but annotations were not included in the training scheme and the clustering process is unsupervised. After obtaining the dictionary, supervised learning method was used to evaluate the performance of the framework by adding user-recorded data without labels. The experimental results showed that extraneous samples would not impair the system's performance significantly and our framework still outperformed the feature engineering approach. We can see the overall performance in terms of

confusion matrix in Figure 6.6. For example, the occupant activity of using hair dryer could be classified somewhat accurately with an accuracy of approximately 100%. The occupant activity of using kettle was also identified precisely with an accuracy of 97%.

FIGURE 6.6: Overall performance in confusion matrix using user-recorded data when combined with extraneous unlabeled data for dictionary learning using SVM. This result is based on the same conditions as in web-collected dataset.

6.3.4 Discussion on Noise Effect and Limitation

The framework is also validated under different setting of Signal-to-Noise Ratio (SNR). The results are shown in Figure 6.7. It is expected that with a lower SNR, the performance of the audio algorithm would easily be affected. And the above results are indeed decreased to some extent. But the accuracy is still satisfactory, which is approximately around 90%. Actually, it can be enhanced by using some noise suppression methods through both hardware and software. However, as the application is designed for the commercial smartphones, it is unrealistic to put extra add-ons to smartphones. In the implementation of the algorithms, we applied a software technique which is common in the related work [15] that removes the silent segments by calculating the energy of the audio segments and we discard them if they are smaller than a threshold. As the proposed framework did not consider the

overlapped sound events, it is the limitation and the mixed sound event recognition can be further explored in the future work.

FIGURE 6.7: The LLC-based framework using unlabeled acoustic data for different Signal-to-Noise Ratio performances

6.4 Summary

To sum up, we have investigated the utilization of unlabeled data and demonstrated its effectiveness in improving the performance of energy-related sound event recognition by using the LLC approach. The requirement to provide labels can be rather inconvenient to users, so our approach is user-friendly as it can achieve satisfactory results when there are only a few labels. We have demonstrated that using both web-collected and user-recorded unlabeled datasets, our approach could enhance the recognition performances in terms of F_1 -score and accuracy when compared to feature engineering approaches.

In order to further extend the system into broader applications, the recognition performance of the system should be improved. The LLC based framework do not capture distinct patterns in a comprehensive way. That is to say, a better feature learning framework can be further studied to enhance the sound event classification system.

Chapter 7

Energy-related Sound Event Recognition Using Feature Learning

In this chapter, energy-related sound event recognition is further studied by using a feature learning method. In last chapter, a semi-supervised feature learning method is studied, which can make use of unlabeled data. This is meaningful, but the performance still needs improvement. The dictionary learning method do not provide better representative learning patterns, which limits its usages in more applications. As a result, to further broaden the usefulness of the energy-related sound event classification, the recognition performance needs to be enhanced. Here, a framework of randomized neural networks with convolutional receptive fields are presented to process the audio signals recorded by microphones embedded in smartphones to extract the high-level features automatically with minimal domain-expert knowledge.

7.1 Methodology

The framework of energy-related sound event classification is proposed based on ELM-LRF. The architecture of ELM-LRF is shown in Fig. 7.1. The recorded audio files are processed into many frames through window cropping. For each frame,

FIGURE 7.1: Architecture of the ELM-LRF for audio classification using random weights and locally dense connections. (a) Input $\mathbf{x}_n$ is the frame-level spectrogram feature with the size of $d \times d$, and output $\hat{\mathbf{Y}}$ is the probabilistic multi-class value; (b) Local receptive fields: K is the number of local receptive fields for feature mappings, and its size is $r \times r$. The convolution filtered result is $\mathbf{C}$ with the size of $(d-r+1)^2 \times K$; (c) $\mathbf{H}_c$ is the pooling results; (d) β is determined by using ELM. Finally, predicted label is calculated for L classes.

FIGURE 7.2: Data visualization using t-SNE. (Better seen in color)

we can compute the Short Time Fourier Transform (STFT) to obtain the time-frequency representation as a 2-dimensional image. Traditionally, typical feature engineering methods would analyze the auditory images based on gradients and other characteristics. Here, engineered feature is not needed, but a feature mapping is performed using local receptive fields with random weights and connections to learn high-level representations. That is to say, minimal expert knowledge is required by the system. The learned high-level representations are then processed by a standard randomized neural networks as the classifier. Thus, the proposed architecture is divided into two steps: transforming the data and determining the output weights. Note that in this proposed architecture, hyperparameters are far fewer than existing methods. These include the number of receptive fields and their sizes, pooling function types and their sizes. The typical hyperparameters in deep learning are the learning rate, the number of epochs and the methods to solve the optimization problem. The proposed framework would not include these parameters, thereby removing the burden to tune them. The data after transformation is visualized in Fig. 7.2 by using t-SNE [158].

We note that the sound of an electrical appliance is relatively long, and is non stationary and transient in nature. However, in our approach, we require the assumption of short-term stationary. To meet the required short-term stationary property, the audios are cropped into frames with all silent frames removed. That is, the acoustic sound is divided into thousands of short-time frames. Hence, the audios are processed at the frame-level so as to meet the short-term stationary assumption. The baseline approach, i.e. the method of MFCCs with GMM used in the DCASE competition, is also based on this short-term stationary process.

7.2 Experiments and Discussions

In this section, the performance results on two datasets are reported and the proposed architecture is compared with baseline methods and other feature learning approaches.

TABLE 7.1: Summary of frame-level performance of energy-related testing dataset in terms of frame level on CREATE recorded dataset.

ID	Classes	Windows	Precision
1	coffee machine	400	100.00%
2	fan	400	88.25%
3	kettle	400	94.75%
4	personal computer	400	79.00%
5	printer	400	93.25%
6	flushing toilet	400	86.75%
7	water tap	400	93.75%
8	hair dryer	400	92.75%
9	background noise (no activity)	400	85.50%
Avg.	Total	3600	90.44%

7.2.1 Data Acquisition

Our focus of energy-related sound events are mostly about the sounds of activated electrical appliances and other resource-related appliances.

In order to evaluate the effectiveness of the proposed system for daily life applications, the labeled dataset described in Chapter 6.2.1 is used to evaluate the proposed framework. As it is recorded in the CREATE Tower, Singapore, we call it CREATE recorded dataset here for convenience. All the files can be publicly accessed from the website¹. In total, 8 classes of sound events related to resource utilization plus one class of background noise (or no activity) in the office and living environment were included. The information of the dataset collected at CREATE is shown in Table 7.1.

The proposed architecture is further tested on the public ESC-50 dataset [159], where some activities are related to energy consumption. The information of this dataset is as follows. The sampling rate is 44.1 kHz and the recording is single channel with Vorbis/Ogg compression rate at 192 kbit/s. This dataset consists of 50 classes and each class has 40 files with each file being of 30 seconds duration. The audio sounds range from animal sounds to urban noise. But only 7 classes were used for the purpose of the evaluating our proposed system, and these are: **water dropping, toilet flushing, brushing teeth, mouse click, keyboard typing, washing machine and vacuum cleaner.**

¹https://drive.google.com/file/d/0B6KX6IV_47RyUnZMcFB2YnR3UE0/view?usp=sharing

7.2.2 Implementation Details

Each audio file was cropped into frames using a window length of 25ms. The overlapping rate was 50%. Then, STFT was applied to obtain the spectrogram, and a logarithm operation was used to scale the raw frequency domain features. If the value of the spectrogram of a frame was less than a predetermined threshold, it would be removed because it is probably a silent frame. For the compared methods, the Mel bank was applied on the spectrogram so as to get the Mel-spectrogram. Finally for the feature engineering methods, Discrete Cosine Transform was used to obtain the MFCCs features with 24 coefficients. The 1st order difference (1-delta) and the 2nd order difference (2-delta) of MFCCs were applied. These designed features were concatenated. We used the Matlab toolbox rastamat² to obtain the aforementioned features.

Spectrogram was used for the ELM-LRF³ to conduct feature mapping and the learning task. Cross validation was applied with the optimal parameters that yield the best results. The parameters were as follows: the receptive size $r \times r = 4 \times 4$, the number of feature maps $K = 48$, the pooling size $e = 3$, and the regularization factor $\lambda = 0.001$. In addition, CRVFL⁴ is also compared and used as the feature and classifier learning tool.

The baseline method was based on MFCCs with the 1-delta and 2-delta features before feeding to Gaussian Mixture Model (GMM) [91], which was the baseline in the DCASE 2016 challenge. The number of mixed components was chosen to be 16. As for the method of Englert *et al.* [15], MFCCs and the delta features were fed into Random Forest (RF). For the dictionary learning method used by Salamon *et al.* [108], the mel-spectrogram was applied to the spherical k-means for coefficients encoding. The number of trees in RF was set to be 500 and the dictionary size is 2048 as stated in their paper. For the deep learning method used by Piczak *et al.* [160], Mel-spectrogram was put into CNN with 2 dimensions (CNN-2D). The parameters followed the codes⁵ they provided. For the deep learning architecture used by Hertel *et al.* [109], spectrogram was used in the 1-dimensional CNN (CNN-1D) architecture to learn high-level representations. The structure followed their

²<http://labrosa.ee.columbia.edu/matlab/rastamat/>

³<https://github.com/ExtremeLearningMachines/ELM-LRF>

⁴<https://sites.google.com/site/zhangleuestc/>

⁵<https://github.com/karoldvl/paper-2015-esc-convnet>

TABLE 7.2: Comparison of classification accuracy based on CREATE recorded dataset. The testing results are reported in the format of average accuracy (its standard variance). The 1-delta and 2-delta denote the 1st-order and 2nd-order difference of MFCCs.

Literature	Features	Methods	Testing(%)	Time(s)
Baseline [91]	MFCCs, 1-delta, 2-delta	GMM	77.64(± 0.46)	16
F. Englert <i>et al.</i> [15]	MFCCs, 1-delta, 2-delta	RF	88.65(± 0.16)	407
J. Salamon <i>et al.</i> [108]	Mel-spectrogram	Spherical Kmeans + RF	81.16(± 4.08)	2767
CRVFL [26]	Spectrogram	CRVFL	88.38(± 0.01)	73
Proposed architecture	Spectrogram	ELM-LRF	88.11(± 0.48)	119

paper with 4 convolutional and pooling layers. Additionally, dropout layers were also included. The parameters were set as stated in their paper. The computing machine was Intel Xeon CPU 2.70 (GHz) with Tesla K40c GPU, and a RAM of 512 GB.

7.2.3 Experimental Results

The proposed architecture was first evaluated with the daily life dataset collected at CREATE. Five-fold cross validation was utilized to tune the hyperparameters of the proposed method. Specifically, all training data was divided into 5 folds, and experiments were run for 5 times. Each time, one of the five folds was selected as the validation set, and the rest was used as the training set. Then, frame-level classification performance was reported on the testing data as tabulated in Table 7.1 so as to be consistent with results in the previous work for comparison [15]. Comparisons with baseline methods, dictionary and deep learning methods were conducted. The performance evaluations of the state-of-the-arts are shown in Table 7.2. The baseline with MFCCs and their corresponding delta features using GMM [91] method achieved a 77.64% testing accuracy. But it was the fastest among all the approaches. Englert *et al.* [15] applied the same features as baseline and RF was reported to be one of the best classifiers. Their approach could obtain the best testing result of as high as 88.65%. Salamon *et al.* [108] applied the spherical K-means to cluster the Mel-spectrogram features, followed by RF as the classifier. Although their performance with the training data could be as high as 99.94%, it seemed to be overfitting in terms of testing accuracy as seen by a large variance. The computational time of the dictionary method was the highest. We could find that the proposed architecture could achieve a satisfactory result at a testing accuracy of 88.11%, which was approximately 10% better than

the baseline methods. Although it was not the best, it still achieved competitive results in an efficient manner, about $4\times$ speedup as compared to Englert’s approach [15]. The confusion matrix of the proposed architecture is shown in Fig. 7.3. The performance of CRVFL based on the spectrogram using the feature learning method is satisfactory when compared to these experimental results. As a result, the randomized neural networks used as feature learning are demonstrated to be effective in recognizing these sound events.

FIGURE 7.3: Confusion matrix of proposed architecture on CREATE recorded dataset with 9 classes. The ID is consistent with Table 7.1.

As for the evaluation of the proposed architecture on the ESC-50 dataset⁶ [159], five-fold cross validation was performed using the training data to tune the parameters of the proposed system. The classification performance is summarized in Table 7.3. Note that file-level results are reported here for the purpose of fair comparison and to be consistent with the work by Piczak. et al [160]. The baseline method obtained a mid-level testing accuracy of 68.21% but it was the fastest. The approach of Englert *et al.* [15] could provide good performance with the training data, and the testing result had about 77.14% accuracy, as good as their performance in the

⁶<https://github.com/karoldvl/ESC-50>

TABLE 7.3: Comparison of classification accuracy in terms of file-level is based on the ESC-50 dataset. The testing results are reported in the format of average accuracy (its standard variance). The 1-delta and 2-delta features denote the 1st-order and 2nd-order difference of MFCCs. CNN-1D and CNN-2D stand for Convolutional Neural Network with 1 and 2 dimensional inputs, respectively.

Literature	Features	Methods	Testing(%)	Time(s)
Baseline [91]	MFCCs, 1-delta, 2-delta	GMM	68.21(± 3.19)	535
F. Englert <i>et al.</i> [15]	MFCCs, 1-delta, 2-delta	RF	77.14(± 2.33)	3725
J. Salamon <i>et al.</i> [108]	Mel-spectrogram	Spherical Kmeans + RF	66.07(± 5.65)	22658
L. Hertel <i>et al.</i> [109]	Spectrogram	CNN-1D	50.53(± 1.27)	1200
K. J. Piczak[160]	Mel-spectrogram	CNN-2D	65.66(± 2.31)	2100
CRVFL[26]	Spectrogram	CRVFL	79.29(± 1.60)	656
Proposed architecture	Spectrogram	ELM-LRF	81.07(± 2.99)	938

previous dataset. The dictionary learning method yielded overfitting results. As the size of the data was increased, it took longer to be optimized in the clustering process. As for deep learning methods, both of the CNN methods did not show satisfactory results. It seemed that CNN-1D [109] was overfitting. It took about 30 epochs, with each epoch taking an average of 40 seconds. It took even longer for the CNN-2D method [160], 300 epochs exactly. The testing accuracy was 65.66%, which was somewhat close to the full data results shown in their paper using short frames and a majority voting scheme. We could see that our proposed architecture achieved a good testing accuracy of 81.07%, which was about 12% better than the baseline methods. It achieved the best performance among these 6 methods. We however have the following remarks on the gap between training and testing accuracies of some methods. We note that the outcomes of the experiments of some methods may be overfitting. The possible reason is that the default settings of the other methods were used. The default settings were tuned in other datasets with some tricks, which was difficult for us to tune them using random search in the hyper-parameter space as the computational time would be too long. For example, it costs almost several hours using the method proposed by J. Salamon *et al.* [108] to obtain an outcome under a hyperparameter setting. As a result, the overfitting phenomenon may happen with the ESC-50 dataset. As for the running time, the proposed architecture was also one of the fastest among these methods. Based on experimental results, we could conclude that the proposed architecture can achieve the best results, and yet computationally efficient. The compared experiment of CRVFL shows satisfactory performance when compared these methods on the ESC-50 energy-related dataset. The performance improvement of these randomized neural networks used as feature learning is significant.

TABLE 7.4: Comparison of the feature learning structures in audio classification. CNN is the Convolutional Neural Network, and 1D and 2D are the input dimensions. BP is short for Back Propagation. MLP denotes the Multilayer perceptron and RF is short for Random Forest. LRF is short for Local Receptive Fields and NN is short for neural networks.

Literature	Feature mapping	Iterative	Layers	Filter number	Parameters	Classifier
L. Hertel <i>et al.</i> [109]	CNN-1D	Yes (BP)	4	(48,96,192,384)	1,513,351	MLP
K. J. Piczak[160]	CNN-2D	Yes (BP)	2	(80,80)	8,206,320	MLP
J. Salamon <i>et al.</i> [108]	Spherical K-means	Yes	1	2048	81,920	RF
Proposed architecture	LRF	No (Closed-form)	1	48	40,368	Random NN

Through the two datasets evaluated, the proposed architecture can handle both small and large scale datasets. Unlike the deep learning methods which require big data to determine the weights of the neural network, the random weights and connections of our proposed architecture are able to learn the high-level representations automatically for both small and large scale data. Besides, the closed-form solution makes the proposed architecture a good candidate for real time applications. Furthermore, far fewer hyperparameters need to be tuned, indicating that minimal expert knowledge is required. Last but not least, no feature engineering is required. The comparison of the structures and methods between the proposed architecture and the other state-of-the-arts is summarized in Table 7.4.

7.2.4 Sound Detection

To evaluate the practical realizability of the proposed system, an experiment was also conducted for the detection of no-activity, i.e. background noise, as the ninth class. The no activity class was recorded in a quiet office environment with only one or two staff in the office. By adding the extra null class, we could regard the energy-related sound activity detection as a classification problem by distinguishing the no-activity class from other activities. If the classifier could distinguish the no-activity class from the aforementioned energy-related sound activities, the proposed system is deemed to be somewhat realistic in terms of sound detection. The classification performance after adding the additional no activity class is tabulated in Table 7.1. From this table, it is clear that the proposed system could achieve a high 85.50% performance. Therefore, the proposed system could effectively distinguish the difference between the background environment and the other energy-related sound events. Besides, in the confusion matrix shown in Fig. 7.3, the background class, indexed as 9, could also be distinguished readily from the other classes.

7.2.5 Robustness against Noise

The robustness of the proposed system was also examined against Additive White Gaussian Noise under different Signal-to-Noise Ratio (SNR) conditions, namely 20, 10, 6, 3 and 0dB. The performance results are shown in Fig. 7.4. It is expected that with noise added to the audios, the performance of the systems would deteriorate. Indeed, all the four systems, including our proposed system, were affected by noise. However, our proposed system and the one by F. Englert *et al.* [15] still achieved satisfactory results under low SNR conditions. For example, with an SNR of 10dB, our proposed system could achieve approximately an accuracy of 86.58% whereas the system by F. Englert *et al.* [15] achieved 86.81%. With this robust performance against additive noise, it somewhat demonstrated the generalization and practical applicability of the proposed system.

FIGURE 7.4: Performance comparison with Additive White Gaussian Noise.

7.2.6 Overlapping Events

The proposed architecture is designed for single energy-related sound event classification, and it seems that it cannot tackle the problem of overlapping sound event classification, neither can those compared methods handle this problem. That is, a multi-class single label classifier cannot handle the multi-class multi-label problem.

Indeed, as discussed in the DCASE competition [91], the task of synthetic sound detection of overlapping events was separated from that of single label sound event classification. The current proposed system can only be applied to the 1st task of the DCASE challenge, i.e. single sound event classification. However, we believe that the proposed system may be further developed for the 2nd task of the DCASE challenge, i.e. overlapping task, by using a multi-label classifier or model.

7.3 Summary

A framework with the ELM-LRF has been proposed to solve the energy-related sound event classification problem. The proposed architecture is able to learn distinctive features automatically from the raw frequency domain data and no engineered feature is required hence minimal expert knowledge is needed when using the proposed framework. We want to highlight that the closed-form solution of the proposed framework makes it faster than other feature learning methods. Experimental results and performance evaluation based on smartphone recorded dataset and public ESC-50 dataset have demonstrated that the proposed architecture could achieve satisfactory results in an efficient manner as compared to the state-of-the-art methods. Furthermore, the performance under noisy environment has also been demonstrated to be effective.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this thesis, building occupant sensing is studied in two aspects: occupancy level prediction and behavior recognition. Occupant behavior recognition is further divided into simple motion-related activity recognition and energy-related sound event recognition. Specifically, we have proposed four feature extraction frameworks with different sensing modalities. The essential feature extraction techniques for these frameworks are based on feature learning methods to extract high-level representations, which have local dependency and scale invariant properties. Their abilities have been demonstrated to enhance the classification performance.

On the problem of occupancy level prediction, a framework based on neural networks with convolutional calculations is proposed to deal with both the time and frequency domains data collected through environmental sensing. Analysis of the pros and cons of the various sensing modalities indicates that environmental sensing is a viable and preferred non-intrusive technique to capture the occupancy level in normal working days in offices or laboratories. Through environmental sensing, the signals of the CO_2 concentration, temperature, relative humidity and air pressure are collected. These signals do carry valuable information about the occupancy level. But existing models cannot directly utilize these raw signals. Typically, these raw signals are calculated based on statistics. These features work well but if the performance of the system needs improving, it is difficult to design

features without domain-expert knowledge. Consequently, feature learning framework is proposed to automatically extract representative features embedded in the temporal and spectral domains. In this manner, minimal domain-expert knowledge is needed. Neural networks with LRF are effectively applied to extract high-level features because they can obtain representations with local dependency and scale invariant properties. We also found that the combined time and frequency domain raw data could enrich the feature learning process, enhancing the classification performance significantly in terms of occupancy levels, i.e. zero, low, medium and high levels. The proposed feature learning framework using only the time-domain raw-data as inputs could achieve an accuracy improvement of approximately 2% when compared to feature engineering methods. But with combined time and frequency domain data as inputs, the proposed feature learning framework could further achieve accuracy improvement of approximately 4%. Besides, the training process could be realized in real time because of the non-iterative process. To sum up, the proposed feature learning framework with environmental sensing can indeed enhance the performance of occupancy level prediction. The advantages of the proposed framework are that the installed sensors are non-intrusive and the performance is satisfactory in an automatically way. But the disadvantages of the proposed framework are that the environmental parameters are sensitive to the indoor atmosphere and they need a suitable setting with some constraints.

On occupant behavior recognition, simple motion-related activity recognition is studied based on two feature learning methods. Based on inertial sensing, a semi-supervised deep learning method is proposed to handle the time-series signals and to improve the classification performance by utilizing unlabeled data in a temporal ensemble manner. This is very useful because the data on the occupant's physical behaviors is varied and pervasive, but annotations of the data are scarce and time-consuming. How to leverage unlabeled data to improve the classifier is therefore a meaningful problem to explore. A method of deep recurrent neural networks is proposed to extract high-level features that are able to capture the temporal dependency. With data augmentation and dropout technique, the outputs of deep neural networks are stochastic variables and can be integrated in an ensemble manner to evaluate the unsupervised loss. By combining both the supervised and unsupervised losses, the back propagation through time can be utilized to determine the numerous weights in the neural networks. As a result, the performance of the recognition system can be improved especially in the case when labeled data is scarce

but various unlabeled data is abundantly available. When compared with other traditional semi-supervised learning methods, the proposed approach outperformed these methods by about 2 ~ 4% in the public dataset of inertial measurement units. The contribution of the proposed method is that it presents a novel integration of existing algorithms but demonstrated to be effective in a realistic problem, i.e. semi-supervised learning problem with automatically feature learning. To realize occupant behavior recognition based on smartphones, we proposed a framework based on dictionary learning method to identify occupant's simple motion-related activities from raw data recorded by embedded inertial measurement units. The dictionary learning method studied is based on locality-constrained linear coding, which can learn high-level features with locally smooth property. The optimization process is non-iterative, therefore giving an analytical solution to the simple motion-related activity recognition problem. This is very attractive especially when considering possible implementation with the resource-limited smart devices. The experimental results demonstrated that simple motion-related activities can be identified accurately with the proposed framework. The advantage of the proposed framework is that it is based on feature learning without too much expert knowledge but the disadvantage is that it needs further signal processing techniques to achieve a robust solution.

Another occupant behavior recognition studied in this thesis is energy-related sound event recognition. Two recognition frameworks are proposed based on the approaches for occupancy level prediction and simple motion-related activity recognition, but processed in a different manner. Dictionary learning based framework is proposed to deal with unlabeled data crawled from the Internet. This framework has been demonstrated to be efficient with real experimental data. The advantage of the proposed framework is that it can leverage extensive unlabeled data in a feature learning way, but the disadvantage is that it cannot deal with mixed sound classification problem. Another framework based on neural networks with local receptive fields is also proposed and this is based on the two-dimensional spectrogram of audio signals as inputs. The feature learning framework is fast and efficient because of the non-iterative property when compared with existing feature engineering methods. The energy-related sound events that we studied are the audio signals of appliances that are commonly used in offices. The traditional method to extract audio features is to use MFCCs, but the feature learning framework can extract more valuable information based on the spectrogram of audios.

Experimental results revealed that local receptive fields could extract the local dependency from the frequency domain, and they are able to distinguish different categories among interactive appliances. The performance of the proposed system is approximately 12% better than the baseline method and 4% better than the state-of-the-art techniques. Furthermore, its training process is much faster than deep learning methods. The advantage of the proposed framework is that the sound event classification system can be fast and effective, but the disadvantage is that it cannot perform very well in a robust way, e.g. it has yet to be effectively demonstrated for noisy or mixed sound situations.

8.2 Future Work

8.2.1 Integration of Occupant Information for Energy Simulations and Experiments

In future work, we can integrate the obtained occupant information into simulation packages or real experiments to study the problems of energy consumption management and thermal comfort. The simulation software, e.g. EnergyPlus, has some parameters related to occupants as inputs to the entire simulated environments. For example, one of the most important parameters related to users is the actual number of occupants. Besides, the clothing factor is also the occupant related information. The occupants' behaviors are closely related to the metabolism rates, which could be taken into consideration in simulations. Experiments can also be conducted in practice to study the differences between theoretical simulations and real situations.

8.2.2 Indoor Localization

The identification of the actual indoor location of the occupant is another useful information to be studied. By combining the occupants' information with their locations, the artificial intelligent system can provide localized cooling or lighting functionality to avoid unnecessary wastage of energy and to optimize the thermal

comfort level. How to obtain accurate indoor location information with less expensive and non-intrusive techniques is still an unsolved problem and if solved will have significant values in smart buildings and other autonomous logistics applications.

The Channel State Information (CSI) that is based on WiFi is a promising technique to achieve effective indoor localization with high precision and low intrusiveness. The CSI can be obtained easily by using off-the-shelf network integrated chips based on the open source library [161]. As WiFi signals are ubiquitous in buildings, indoor localization based on WiFi is of great interest because of its non-intrusive property. However, traditional methods using fingerprinting based on RSSI could only achieve an accuracy of 2m, which is still not precise enough for some applications. As CSI can deal with the Non-light-of-sight scenario, fingerprinting methods with CSI would be much more precise than those with RSSI. Some recent works have explored the feasibility of leveraging CSI with deep learning methods, and have found promising results [162]. We believe that accurate and non-intrusive indoor localization can be achieved by incorporating the CSI information.

8.2.3 Other Sensing Techniques

Other occupant sensing techniques should also be investigated. For example, one can explore occupant sensing using radio frequency signals. These signals carry much valuable information, and they are also non-intrusive sensing solutions [163]. The radio frequency signals have been used to perform sleep detection [164], multiple person tracking [165] and emotion recognition [166]. Various information of the occupants can be mined by advanced techniques. In addition, their potential applications are not limited to smart building applications, but they can be further extended to applications in health care and others.

8.2.4 Realistic and General Applications

Some of the applications discussed in this thesis can be further enhanced by considering some realistic issues and to be a generalized solution to the mobile internet era. The different categories of the simple motion-related activity recognition can be studied to provide abundant profiles of the smartphone users, so as to increase

the accuracy of the advertisements and intelligent recommending system. For example, it is useful to identify the user content, i.e. the transportation modes or some simple smartphone interactions in the situations of hand holding, walking, sitting or laying down. Another important issue related to the sound event recognition that should be further studied is robustness. To be specific, the mixed sound classification should be studied to increase its robustness, so as to remove some restrictions of the intelligent system.

As for realistic applications, powerful machine learning tools can also be studied in occupant sensing area. For example, Spiking neural networks can be studied as it is regarded as the next generation artificial neural networks. The spiking signals and schemes have been found some similarities in biological phenomena [114]. Besides, transfer learning is also a tool that deserves to be explored in occupant activity recognition.

Appendix A

Codes

A.1 Codes of ELM

Thanks to the author, Hongming Zhou, the codes are provided and revised as follows:

```
function [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy] = ELM_c(
    TrainingData_File, TestingData_File, Elm_Type, NumberofHiddenNeurons,
    ActivationFunction)

% Usage: elm(TrainingData_File, TestingData_File, Elm_Type, NumberofHiddenNeurons,
    ActivationFunction)
% OR:      [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy] = elm(
    TrainingData_File, TestingData_File, Elm_Type, NumberofHiddenNeurons,
    ActivationFunction)
%
% Input:
% TrainingData_File      - Filename of training data set
% TestingData_File       - Filename of testing data set
% Elm_Type               - 0 for regression; 1 for (both binary and multi-classes)
                        classification
% NumberofHiddenNeurons - Number of hidden neurons assigned to the ELM
% ActivationFunction      - Type of activation function:
%                           'sig' for Sigmoidal function
%                           'sin' for Sine function
%                           'hardlim' for Hardlim function
%                           'tribas' for Triangular basis function
%                           'raddbas' for Radial basis function (for additive type
                        of SLFNs instead of RBF type of SLFNs)
%
% Output:
% TrainingTime           - Time (seconds) spent on training ELM
% TestingTime            - Time (seconds) spent on predicting ALL testing data
% TrainingAccuracy       - Training accuracy:
```

```

%                                RMSE for regression or correct classification rate for
    classification
% TestingAccuracy              - Testing accuracy:
%                                RMSE for regression or correct classification rate for
    classification
%
% MULTI-CLASSE CLASSIFICATION: NUMBER OF OUTPUT NEURONS WILL BE AUTOMATICALLY SET
    EQUAL TO NUMBER OF CLASSES
% FOR EXAMPLE, if there are 7 classes in all, there will have 7 output
% neurons; neuron 5 has the highest output means input belongs to 5-th class
%
% Sample1 regression: [TrainingTime, TestingTime, TrainingAccuracy,
    TestingAccuracy] = elm('sinc_train', 'sinc_test', 0, 20, 'sig')
% Sample2 classification: elm('diabetes_train', 'diabetes_test', 1, 20, 'sig')
%
%    %%%    Authors:    MR QIN-YU ZHU AND DR GUANG-BIN HUANG
%    %%%    NANYANG TECHNOLOGICAL UNIVERSITY, SINGAPORE
%    %%%    EMAIL:    EGBHUANG@NTU.EDU.SG; GBHUANG@IEEE.ORG
%    %%%    WEBSITE:    http://www.ntu.edu.sg/eee/icis/cv/egbhuang.htm
%    %%%    DATE:    APRIL 2004

%%%%%%%%%%%% Macro definition
REGRESSION=0;
CLASSIFIER=1;

%%%%%%%%%%%% Load training dataset
train_data=TrainingData_File;
T=train_data(:,1)';
P=train_data(:,2:size(train_data,2))';
clear train_data;                                % Release raw training data
    array

%%%%%%%%%%%% Load testing dataset
test_data=TestingData_File;
TV.T=test_data(:,1)';
TV.P=test_data(:,2:size(test_data,2))';
clear test_data;                                % Release raw testing data
    array

NumberOfTrainingData=size(P,2);
NumberOfTestingData=size(TV.P,2);
NumberOfInputNeurons=size(P,1);

if Elm_Type~=REGRESSION
    %%%%%%%%%%%%% Preprocessing the data of classification
    sorted_target=sort(cat(2,T,TV.T),2);
    label=zeros(1,1);                                % Find and save in 'label'
    class label from training and testing data sets
    label(1,1)=sorted_target(1,1);
    j=1;
    for i = 2:(NumberOfTrainingData+NumberOfTestingData)
        if sorted_target(1,i) ~= label(1,j)
            j=j+1;
            label(1,j) = sorted_target(1,i);
        end
    end
end

```

```

        end
    end
    number_class=j;
    NumberofOutputNeurons=number_class;

    %%%%%%%%% Processing the targets of training
    temp_T=zeros(NumberofOutputNeurons, NumberofTrainingData);
    for i = 1:NumberofTrainingData
        for j = 1:number_class
            if label(1,j) == T(1,i)
                break;
            end
        end
        temp_T(j,i)=1;
    end
    T=temp_T*2-1;

    %%%%%%%%% Processing the targets of testing
    temp_TV_T=zeros(NumberofOutputNeurons, NumberofTestingData);
    for i = 1:NumberofTestingData
        for j = 1:number_class
            if label(1,j) == TV.T(1,i)
                break;
            end
        end
        temp_TV_T(j,i)=1;
    end
    TV.T=temp_TV_T*2-1;

end % end if of Elm_Type

%%%%%%%% Calculate weights & biases
start_time_train=cputime;

%%%%%%%% Random generate input weights InputWeight ( $w_i$ ) and biases
    BiasofHiddenNeurons ( $b_i$ ) of hidden neurons
    InputWeight=rand(NumberofHiddenNeurons, NumberofInputNeurons)*2-1;
    BiasofHiddenNeurons=rand(NumberofHiddenNeurons,1);
    tempH=InputWeight*P;
    clear P; % Release input of training
    data
    ind=ones(1, NumberofTrainingData);
    BiasMatrix=BiasofHiddenNeurons(:,ind); % Extend the bias matrix
    BiasofHiddenNeurons to match the demention of H
    tempH=tempH+BiasMatrix;

%%%%%%%% Calculate hidden neuron output matrix H
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%%% Sigmoid
        H = 1 ./ (1 + exp(-tempH));
    case {'sin','sine'}
        %%%%%%%%% Sine

```

```

        H = sin(tempH);
    case {'hardlim'}
        %%%%%%%%% Hard Limit
        H = double(hardlim(tempH));
    case {'tribas'}
        %%%%%%%%% Triangular basis function
        H = tribas(tempH);
    case {'radbas'}
        %%%%%%%%% Radial basis function
        H = radbas(tempH);
        %%%%%%%%% More activation functions can be added here
end
clear tempH; % Release the temporary
array for calculation of hidden neuron output matrix H

%%%%%%%%%% Calculate output weights OutputWeight (beta_i)
%OutputWeight=pinv(H') * T'; % implementation without
regularization factor //refer to 2006 Neurocomputing paper
lambda = eig(H*H');
C = max(lambda);%50;
%OutputWeight=inv(eye(size(H,1))/C+H * H') * H * T'; % faster method 1 //refer
to 2012 IEEE TSMC-B paper
%implementation; one can set regularizaiton factor C properly in classification
applications

OutputWeight=(eye(size(H,1))/C+H * H') \ H * T'; % faster method 2 //refer to
2012 IEEE TSMC-B paper
%implementation; one can set regularizaiton factor C properly in classification
applications

%If you use faster methods or kernel method, PLEASE CITE in your paper properly:

%Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang, "Extreme Learning
Machine for Regression and Multi-Class Classification," submitted to IEEE
Transactions on Pattern Analysis and Machine Intelligence, October 2010.

end_time_train=cputime;
TrainingTime=end_time_train-start_time_train % Calculate CPU time (
seconds) spent for training ELM

TrainingTime=1;
%%%%%%%%%% Calculate the training accuracy
Y=(H' * OutputWeight)'; % Y: the actual output of
the training data
if Elm_Type == REGRESSION
    TrainingAccuracy=sqrt(mse(T - Y)) % Calculate training
accuracy (RMSE) for regression case
end
clear H;

%%%%%%%%%% Calculate the output of testing input
start_time_test=cputime;

tempH_test=InputWeight*TV.P;

```

```

clear TV.P; % Release input of testing data
ind=ones(1,NumberOfTestingData);
BiasMatrix=BiasofHiddenNeurons(:,ind); % Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH_test=tempH_test + BiasMatrix;
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%%% Sigmoid
        H_test = 1 ./ (1 + exp(-tempH_test));
    case {'sin','sine'}
        %%%%%%%%% Sine
        H_test = sin(tempH_test);
    case {'hardlim'}
        %%%%%%%%% Hard Limit
        H_test = hardlim(tempH_test);
    case {'tribas'}
        %%%%%%%%% Triangular basis function
        H_test = tribas(tempH_test);
    case {'radbas'}
        %%%%%%%%% Radial basis function
        H_test = radbas(tempH_test);
        %%%%%%%%% More activation functions can be added here
end
TY=(H_test' * OutputWeight)'; % TY: the actual output of
the testing data
TV_temp=TV.T;
save TY_out TY TV_temp
%save TV_out TV_temp
end_time_test=cputime;
TestingTime=end_time_test-start_time_test % Calculate CPU time (
seconds) spent by ELM predicting the whole testing data

if Elm_Type == REGRESSION
    TestingAccuracy=sqrt(mse(TV.T - TY)) % Calculate testing accuracy
    (RMSE) for regression case
end

if Elm_Type == CLASSIFIER
    %%%%%%%%% Calculate training & testing classification accuracy
    MissClassificationRate_Training=0;
    MissClassificationRate_Testing=0;

    for i = 1 : size(T, 2)
        [x, label_index_expected]=max(T(:,i));
        [x, label_index_actual]=max(Y(:,i));
        if label_index_actual~=label_index_expected
            MissClassificationRate_Training=MissClassificationRate_Training+1;
        end
    end
    TrainingAccuracy=1-MissClassificationRate_Training/size(T,2)
    for i = 1 : size(TV.T, 2)
        [x, label_index_expected]=max(TV.T(:,i));
        [x, label_index_actual]=max(TY(:,i));
        if label_index_actual~=label_index_expected

```

```

        MissClassificationRate_Testing=MissClassificationRate_Testing+1;
    end
end
TestingAccuracy=1-MissClassificationRate_Testing/size(TV.T,2)
end

```

A.2 Codes of RVFL

Thanks to the author, Le Zhang, the codes are provided and revised as follows:

```

function [train_predict,train_accuracy,test_predict,test_accuracy]=
    RVFL_train_classify(trainX,trainY,testX,testY,option)
% This is the function to train and evaluate RVFL for classification
% problem.
% Option.N :          number of hidden neurons
% Option.bias:        whether to have bias in the output neurons
% option.link:         whether to have the direct link.
% option.ActivationFunction: Activation Functions used.
% option.seed:         Random Seeds
% option.mode          1: regularized least square, 2: Moore-Penrose pseudoinverse
% option.RandomType: different randomization methods. Currently only support
    Gaussian and uniform.
% option.Scale         Linearly scale the random features before feedinto the nonlinear
    activation function.
%
    In this implementation, we consider the threshold which lead to
    0.99 of the maximum/minimum value of the activation function as the saturating
    threshold.
%
    Option.Scale=0.9 means all the random features will be linearly
    scaled
%
    into 0.9* [lower_saturating_threshold,upper_saturating_threshold
    ].
%option.Scalemode Scalemode=1 will scale the features for all neurons.
%
    Scalemode=2 will scale the features for each hidden
%
    neuron separately.
%
    Scalemode=3 will scale the range of the randomization for
%
    uniform diatribution.

% This software package has been developed by Le Zhang(c) 2015
% based on this paper: A Comprehensive Evaluation of Random Vector Functional Link
    Neural Network Variants
% For technical support and/or help, please contact Lzhang027@e.ntu.edu.sg
% This package has been downloaed from https://sites.google.com/site/zhangleuestc/
if ~isfield(option,'N')|| isempty(option.N)
    option.N=100;
end
if ~isfield(option,'bias')|| isempty(option.bias)
    option.bias=false;
end
if ~isfield(option,'link')|| isempty(option.link)
    option.link=true;

```

```

end
if ~isfield(option,'ActivationFunction') || isempty(option.ActivationFunction)
    option.ActivationFunction='radbas';
end
if ~isfield(option,'seed') || isempty(option.seed)
    option.seed=0;
end
if ~isfield(option,'RandomType') || isempty(option.RandomType)
    option.RandomType='Uniform';
end
if ~isfield(option,'mode') || isempty(option.mode)
    option.mode=1;
end

end
if ~isfield(option,'Scale') || isempty(option.Scale)
    option.Scale=1;
end

end
if ~isfield(option,'Scalemode') || isempty(option.Scalemode)
    option.Scalemode=1;
end
end
rand('state',option.seed);
randn('state',option.seed);
U_trainY=unique(trainY);
nclass=numel(U_trainY);
trainY_temp=zeros(numel(trainY),nclass);
% 0-1 coding for the target
for i=1:nclass
    idx= trainY==U_trainY(i);

    trainY_temp(idx,i)=1;
end
% trainY_temp = trainY; % for regression
[Nsample,Nfea]=size(trainX);
N=option.N;
if strcmp(option.RandomType,'Uniform')
    if option.Scalemode==3
        Weight= option.Scale*(rand(Nfea,N)*2-1);
        Bias= option.Scale*rand(1,N);
        % fprintf('linearly scale the range of uniform distribution to %d\n',
        option.Scale);
    else
        Weight=rand(Nfea,N)*2-1;
        Bias=rand(1,N);
    end
else if strcmp(option.RandomType,'Gaussian')
    Weight=randn(Nfea,N);
    Bias=randn(1,N);
else
    error('only Gaussian and Uniform are supported')
end
end
Bias_train=repmat(Bias,Nsample,1);
H=trainX*Weight+Bias_train;

```



```

switch lower(option.ActivationFunction)
    case {'sig','sigmoid'}

        if option.Scale

            Saturating_threshold=[-4.6,4.6];
            Saturating_threshold_activate=[0,1];
            if option.Scalemode==1;

                [H,k,b]=Scale_feature(H,Saturating_threshold,option.Scale);
            elseif option.Scalemode==2

                [H,k,b]=Scale_feature_separately(H,Saturating_threshold,option.Scale);
            end
        end
        H = 1 ./ (1 + exp(-H));
    case {'sin','sine'}

        if option.Scale

            Saturating_threshold=[-pi/2,pi/2];
            Saturating_threshold_activate=[-1,1];
            if option.Scalemode==1

                [H,k,b]=Scale_feature(H,Saturating_threshold,option.Scale);
            elseif option.Scalemode==2

                [H,k,b]=Scale_feature_separately(H,Saturating_threshold,option.Scale);
            end
        end
        H = sin(H);
    case {'hardlim'}

        H = double(hardlim(H));
    case {'tribas'}

        if option.Scale

            Saturating_threshold=[-1,1];
            Saturating_threshold_activate=[0,1];
            if option.Scalemode==1

                [H,k,b]=Scale_feature(H,Saturating_threshold,option.Scale);
            elseif option.Scalemode==2

                [H,k,b]=Scale_feature_separately(H,Saturating_threshold,option.Scale)
            end
        end
        H = tribas(H);
    case {'radbas'}

        if option.Scale

```

```

    Saturating_threshold=[-2.1,2.1];
    Saturating_threshold_activate=[0,1];
    if option.Scalemode==1

        [H,k,b]=Scale_feature(H,Saturating_threshold,option.Scale);
    elseif option.Scalemode==2

        [H,k,b]=Scale_feature_separately(H,Saturating_threshold,option.Scale);
    end
end
H = radbas(H);

case {'sign'}
    H = sign(H);
end
if option.bias
    H=[H,ones(Nsample,1)];
end
if option.link

    switch option.Scalemode
        case 1
            trainX_temp=trainX.*k+b;
            H=[H,trainX_temp];
        case 2
            [trainX_temp,ktr,btr]=Scale_feature_separately(trainX,
Saturating_threshold_activate,option.Scale);
            H=[H,trainX_temp];
        otherwise
            H=[H,trainX];
    end

end
H(isnan(H))=0;
if option.mode==2

beta=pinv(H)*trainY_temp;
else if option.mode==1

    if ~isfield(option,'C')||isempty(option.C)
        option.C=0.1;
    end
    C=option.C;
    if N<Nsample
        beta=(eye(size(H,2))/C+H' * H) \ H'*trainY_temp;
    else
        beta=H'*((eye(size(H,1))/C+H* H') \ trainY_temp);
    end
    else
        error('Unsupport mode, only Regularized least square and Moore-Penrose
pseudoinverse are allowed. ')
    end
end
end

```

```

trainY_temp=H*beta;
Y_temp=zeros(Nsample,1);
% decode the target
for i=1:Nsample
    [maxvalue,idx]=max(trainY_temp(i,:));
    Y_temp(i)=U_trainY(idx);
end

Bias_test= repmat(Bias,numel(testY),1);
H_test=testX*Weight+Bias_test;
switch lower(option.ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%%% Sigmoid
        if option.Scale
            if option.Scalemode==1
                H_test=H_test.*k+b;
            elseif option.Scalemode==2
                nSamtest=size(H_test,1);
                kt=repmat(k,nSamtest,1);
                bt=repmat(b,nSamtest,1);
                H_test=H_test.*kt+bt;
            end
        end
        H_test = 1 ./ (1 + exp(-H_test));
    case {'sin','sine'}

        if option.Scale
            if option.Scalemode==1
                H_test=H_test.*k+b;
            elseif option.Scalemode==2
                nSamtest=size(H_test,1);
                kt=repmat(k,nSamtest,1);
                bt=repmat(b,nSamtest,1);
                H_test=H_test.*kt+bt;
            end
        end
        H_test = sin(H_test);
    case {'hardlim'}

        H_test = double(hardlim(H_test));
    case {'tribas'}

        if option.Scale
            if option.Scalemode==1
                H_test=H_test.*k+b;
            elseif option.Scalemode==2
                nSamtest=size(H_test,1);
                kt=repmat(k,nSamtest,1);
                bt=repmat(b,nSamtest,1);
                H_test=H_test.*kt+bt;
            end
        end
        H_test = tribas(H_test);
    case {'radbas'}

```

```

        if option.Scale
            if option.Scalemode==1
                H_test=H_test.*k+b;
            elseif option.Scalemode==2
                nSamtest=size(H_test,1);
                kt= repmat(k,nSamtest,1);
                bt= repmat(b,nSamtest,1);
                H_test=H_test.*kt+bt;
            end
        end
        H_test = radbas(H_test);
        case {'sign'}

            H_test = sign(H_test);

    end

    if option.bias
        H_test=[H_test,ones(numel(testY),1)];
    end
    if option.link
        switch option.Scalemode
            case 1
                testX_temp=testX.*k+b;
                H_test=[H_test,testX_temp];
            case 2
                nSamtest=size(H_test,1);
                kt= repmat(ktr,nSamtest,1);
                bt= repmat(btr,nSamtest,1);
                testX_temp=testX.*kt+bt;
                H_test=[H_test,testX_temp];
            otherwise
                H_test=[H_test,testX];
        end
    end
    end
    H_test(isnan(H_test))=0;
    testY_temp=H_test*beta;
    Yt_temp=zeros(numel(testY),1);

    for i=1:numel(testY)
        [maxvalue,idx]=max(testY_temp(i,:));
        Yt_temp(i)=U_trainY(idx);
    end

    % train_predict = trainY_temp;
    train_predict = Y_temp;
    % test_predict = testY_temp;
    test_predict = Yt_temp;
    train_accuracy=length(find(Y_temp==trainY))/Nsample;

    test_accuracy=length(find(Yt_temp==testY))/numel(testY);

```

end

```
function [Output,k,b]=Scale_feature(Input,Saturating_threshold,ratio)
Min_value=min(min(Input));
Max_value=max(max(Input));
min_value=Saturating_threshold(1)*ratio;
max_value=Saturating_threshold(2)*ratio;
k=(max_value-min_value)/(Max_value-Min_value);
b=(min_value*Max_value-Min_value*max_value)/(Max_value-Min_value);
Output=Input.*k+b;
end
```

```
function [Output,k,b]=Scale_feature_separately(Input,Saturating_threshold,ratio)
nNeurons=size(Input,2);
k=zeros(1,nNeurons);
b=zeros(1,nNeurons);
Output=zeros(size(Input));
min_value=Saturating_threshold(1)*ratio;
max_value=Saturating_threshold(2)*ratio;
for i=1:nNeurons
Min_value=min(Input(:,i));
Max_value=max(Input(:,i));
k(i)=(max_value-min_value)/(Max_value-Min_value);
b(i)=(min_value*Max_value-Min_value*max_value)/(Max_value-Min_value);
Output(:,i)=Input(:,i).*k(i)+b(i);
end
```

%%%%%%%%%%%%%% Kernel Matrix %%%%%%%%%%%%%%%

```
function omega = kernel_matrix(Xtrain,kernel_type, kernel_pars,Xt)
```

```
nb_data = size(Xtrain,1);
```

```
if strcmp(kernel_type,'RBF_kernel'),
    if nargin<4,
        XXh = sum(Xtrain.^2,2)*ones(1,nb_data);
        omega = XXh+XXh'-2*(Xtrain*Xtrain');
        omega = exp(-omega./kernel_pars(1));
    else
        XXh1 = sum(Xtrain.^2,2)*ones(1,size(Xt,1));
        XXh2 = sum(Xt.^2,2)*ones(1,nb_data);
        omega = XXh1+XXh2' - 2*Xtrain*Xt';
        omega = exp(-omega./kernel_pars(1));
    end
elseif strcmp(kernel_type,'lin_kernel')
    if nargin<4,
        omega = Xtrain*Xtrain';
    else
        omega = Xtrain*Xt';
    end
```

```

elseif strcmp(kernel_type,'poly_kernel')
    if nargin<4,
        omega = (Xtrain*Xtrain'+kernel_pars(1)).^kernel_pars(2);
    else
        omega = (Xtrain*Xt'+kernel_pars(1)).^kernel_pars(2);
    end

elseif strcmp(kernel_type,'wav_kernel')
    if nargin<4,
        XXh = sum(Xtrain.^2,2)*ones(1,nb_data);
        omega = XXh+XXh'-2*(Xtrain*Xtrain');

        XXh1 = sum(Xtrain,2)*ones(1,nb_data);
        omega1 = XXh1-XXh1';
        omega = cos(kernel_pars(3)*omega1./kernel_pars(2)).*exp(-omega./
kernel_pars(1));

    else
        XXh1 = sum(Xtrain.^2,2)*ones(1,size(Xt,1));
        XXh2 = sum(Xt.^2,2)*ones(1,nb_data);
        omega = XXh1+XXh2' - 2*(Xtrain*Xt');

        XXh11 = sum(Xtrain,2)*ones(1,size(Xt,1));
        XXh22 = sum(Xt,2)*ones(1,nb_data);
        omega1 = XXh11-XXh22';

        omega = cos(kernel_pars(3)*omega1./kernel_pars(2)).*exp(-omega./
kernel_pars(1));
    end
end

end
end

```

A.3 Codes of Kernel ELM

Thanks to the author, Hongming Zhou, the codes are provided and revised as follows:

```

function [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy, TY]
= elm_kernel(TrainingData_File, TestingData_File, Elm_Type,
    Regularization_coefficient, Kernel_type, Kernel_para)

%%%%%% Macro definition
REGRESSION=0;
CLASSIFIER=1;

```

```

%%%%%%%%%%%% Load training dataset
train_data=TrainingData_File;%load(TrainingData_File);
T=train_data(:,1)';
P=train_data(:,2:size(train_data,2))';
clear train_data; % Release raw training data
array

%%%%%%%%%%%% Load testing dataset
test_data=TestingData_File;%load(TestingData_File);
TV.T=test_data(:,1)';
TV.P=test_data(:,2:size(test_data,2))';
clear test_data; % Release raw testing data
array

C = Regularization_coefficient;
NumberOfTrainingData=size(P,2);
NumberOfTestingData=size(TV.P,2);

if Elm_Type~=REGRESSION
    %%%%%%%%%%%%% Preprocessing the data of classification
    sorted_target=sort(cat(2,T,TV.T),2);
    label=zeros(1,1);
    label(1,1)=sorted_target(1,1);
    j=1;
    for i = 2:(NumberOfTrainingData+NumberOfTestingData)
        if sorted_target(1,i) ~= label(1,j)
            j=j+1;
            label(1,j) = sorted_target(1,i);
        end
    end
    number_class=j;
    NumberOfOutputNeurons=number_class;

    %%%%%%%%%%%%% Processing the targets of training
    temp_T=zeros(NumberOfOutputNeurons, NumberOfTrainingData);
    for i = 1:NumberOfTrainingData
        for j = 1:number_class
            if label(1,j) == T(1,i)
                break;
            end
        end
        temp_T(j,i)=1;
    end
    T=temp_T*2-1;

    %%%%%%%%%%%%% Processing the targets of testing
    temp_TV_T=zeros(NumberOfOutputNeurons, NumberOfTestingData);
    for i = 1:NumberOfTestingData
        for j = 1:number_class
            if label(1,j) == TV.T(1,i)
                break;
            end
        end
        temp_TV_T(j,i)=1;
    end
end

```

```

end
TV.T=temp_TV_T*2-1;

% end if of Elm_Type

end
%ANN_alltrainlabel=T;
%save ANN_alltrainlabel ANN_alltrainlabel

%ANN_alltestlabel=TV.T;

%save ANN_alltestlabel ANN_alltestlabel

%%%%%%%%%% Training Phase %%%%%%%%%%%%%%%
tic;
n = size(T,2);
Omega_train = kernel_matrix(P',Kernel_type, Kernel_para);
OutputWeight=((Omega_train+speye(n)/C)\(T'));
TrainingTime=toc

%%%%%%%%%% Calculate the training output
Y=(Omega_train * OutputWeight)';

%%%%%%%%%% Calculate the output of testing input
tic;
Omega_test = kernel_matrix(P',Kernel_type, Kernel_para,TV.P');
TY=(Omega_test' * OutputWeight)';
TestingTime=toc

%%%%%%%%%% Calculate training & testing classification accuracy

if Elm_Type == REGRESSION
%%%%%%%%%% Calculate training & testing accuracy (RMSE) for regression case
    TrainingAccuracy=sqrt(mse(T - Y))
    TestingAccuracy=sqrt(mse(TV.T - TY))
end

if Elm_Type == CLASSIFIER
%%%%%%%%%% Calculate training & testing classification accuracy
    MissClassificationRate_Training=0;
    MissClassificationRate_Testing=0;

    for i = 1 : size(T, 2)
        [x, label_index_expected]=max(T(:,i));
        [x, label_index_actual]=max(Y(:,i));
        if label_index_actual~=label_index_expected
            MissClassificationRate_Training=MissClassificationRate_Training+1;
        end
    end
    TrainingAccuracy=1-MissClassificationRate_Training/size(T,2)
    for i = 1 : size(TV.T, 2)
        [x, label_index_expected]=max(TV.T(:,i));
        [x, label_index_actual]=max(TY(:,i));
        if label_index_actual~=label_index_expected
            MissClassificationRate_Testing=MissClassificationRate_Testing+1;
        end
    end
end

```



```

end
TestingAccuracy=1-MissClassificationRate_Testing/size(TV.T,2)
end

Testoutput_temp=TY;
save Testoutput_temp Testoutput_temp
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Kernel Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function omega = kernel_matrix(Xtrain,kernel_type, kernel_pars,Xt)

nb_data = size(Xtrain,1);

if strcmp(kernel_type,'RBF_kernel'),
    if nargin<4,
        XXh = sum(Xtrain.^2,2)*ones(1,nb_data);
        omega = XXh+XXh'-2*(Xtrain*Xtrain');
        omega = exp(-omega./kernel_pars(1));
    else
        XXh1 = sum(Xtrain.^2,2)*ones(1,size(Xt,1));
        XXh2 = sum(Xt.^2,2)*ones(1,nb_data);
        omega = XXh1+XXh2' - 2*Xtrain*Xt';
        omega = exp(-omega./kernel_pars(1));
    end

elseif strcmp(kernel_type,'lin_kernel')
    if nargin<4,
        omega = Xtrain*Xtrain';
    else
        omega = Xtrain*Xt';
    end

elseif strcmp(kernel_type,'poly_kernel')
    if nargin<4,
        omega = (Xtrain*Xtrain'+kernel_pars(1)).^kernel_pars(2);
    else
        omega = (Xtrain*Xt'+kernel_pars(1)).^kernel_pars(2);
    end

elseif strcmp(kernel_type,'wav_kernel')
    if nargin<4,
        XXh = sum(Xtrain.^2,2)*ones(1,nb_data);
        omega = XXh+XXh'-2*(Xtrain*Xtrain');

        XXh1 = sum(Xtrain,2)*ones(1,nb_data);
        omega1 = XXh1-XXh1';
        omega = cos(kernel_pars(3)*omega1./kernel_pars(2)).*exp(-omega./
kernel_pars(1));

    else
        XXh1 = sum(Xtrain.^2,2)*ones(1,size(Xt,1));
        XXh2 = sum(Xt.^2,2)*ones(1,nb_data);
        omega = XXh1+XXh2' - 2*(Xtrain*Xt');
    end
end

```

```

        XXh11 = sum(Xtrain,2)*ones(1,size(Xt,1));
        XXh22 = sum(Xt,2)*ones(1,nb_data);
        omega1 = XXh11-XXh22';

        omega = cos(kernel_pars(3)*omega1./kernel_pars(2)).*exp(-omega./
        kernel_pars(1));
    end
end

```

A.4 Codes of KRR

Thanks to the author, Le Zhang, the codes are provided and revised as follows:

```

function [train_predict,train_accuracy,test_predict,test_accuracy]=kkr(trainX,
    trainY,testX,testY,option)

if ~isfield(option, 'kernel') || isempty(option.kernel)
    option.kernel='RBF_kernel'
end
if ~isfield(option, 'C') || isempty(option.C)
    option.C=0.1;
end
C=option.C;

U_trainY=unique(trainY);
nclass=numel(U_trainY);
trainY_temp=zeros(numel(trainY),nclass);
% 0-1 coding for the target
for i=1:nclass
    idx= trainY==U_trainY(i);

    trainY_temp(idx,i)=1;
end
% trainY_temp = trainY; % for regression
[Nsample,Nfea]=size(trainX);

H = trainX;
H(isnan(H))=0;

K = kernel_matrix(H,option.kernel, option.p_value);

beta=((eye(size(K,2))/C+K)\(trainY_temp));

trainY_temp=K*beta;
Y_temp=zeros(Nsample,1);
% decode the target
for i=1:Nsample
    [maxvalue,idx]=max(trainY_temp(i,:));

```

```

        Y_temp(i)=U_trainY(idx);
    end

H_test = testX;
H_test(isnan(H_test))=0;

H_test = kernel_matrix(H,option.kernel, option.p_value, H_test);
H_test = H_test';

testY_temp=H_test*beta;
Yt_temp=zeros(numel(testY),1);

for i=1:numel(testY)
    [maxvalue,idx]=max(testY_temp(i,:));
    Yt_temp(i)=U_trainY(idx);
end

% train_predict = trainY_temp;
train_predict = Y_temp;
% test_predict = testY_temp;
test_predict = Yt_temp;
train_accuracy=length(find(Y_temp==trainY))/Nsample;

test_accuracy=length(find(Yt_temp==testY))/numel(testY);

end

function omega = kernel_matrix(Xtrain, kernel_type, kernel_pars, Xt)

nb_data = size(Xtrain,1);

if strcmp(kernel_type, 'RBF_kernel'),
    if nargin<4,
        XXh = sum(Xtrain.^2,2)*ones(1,nb_data);
        omega = XXh+XXh'-2*(Xtrain*Xtrain');
        omega = exp(-omega./kernel_pars(1));
    else
        XXh1 = sum(Xtrain.^2,2)*ones(1,size(Xt,1));
        XXh2 = sum(Xt.^2,2)*ones(1,nb_data);
        omega = XXh1+XXh2' - 2*Xtrain*Xt';
        omega = exp(-omega./kernel_pars(1));
    end
elseif strcmp(kernel_type, 'lin_kernel')
    if nargin<4,
        omega = Xtrain*Xtrain';
    else
        omega = Xtrain*Xt';
    end
end

```

```

end

elseif strcmp(kernel_type,'poly_kernel')
    if nargin<4,
        omega = (Xtrain*Xtrain'+kernel_pars(1)).^kernel_pars(2);
    else
        omega = (Xtrain*Xt'+kernel_pars(1)).^kernel_pars(2);
    end

elseif strcmp(kernel_type,'wav_kernel')
    if nargin<4,
        XXh = sum(Xtrain.^2,2)*ones(1,nb_data);
        omega = XXh+XXh'-2*(Xtrain*Xtrain');

        XXh1 = sum(Xtrain,2)*ones(1,nb_data);
        omega1 = XXh1-XXh1';
        omega = cos(kernel_pars(3)*omega1./kernel_pars(2)).*exp(-omega./
kernel_pars(1));

    else
        XXh1 = sum(Xtrain.^2,2)*ones(1,size(Xt,1));
        XXh2 = sum(Xt.^2,2)*ones(1,nb_data);
        omega = XXh1+XXh2' - 2*(Xtrain*Xt');

        XXh11 = sum(Xtrain,2)*ones(1,size(Xt,1));
        XXh22 = sum(Xt,2)*ones(1,nb_data);
        omega1 = XXh11-XXh22';

        omega = cos(kernel_pars(3)*omega1./kernel_pars(2)).*exp(-omega./
kernel_pars(1));
    end
end

end

```

Author's Publications

Journal Articles

- **Qingchang Zhu**, Zhenghua Chen, Mustafa K. Masood, Yeng Chai Soh, “Occupancy estimation with environmental sensing via non-iterative LRF feature learning in time and frequency domains,” *Energy and Buildings*, Vol. 141, pp. 125-133, 2017.
- **Qingchang Zhu**, Zhenghua Chen, Yeng Chai Soh, “A Novel Deep Semi-supervised Learning for Human Activity Recognition,” *submitted to IEEE Transactions on Industrial Informatics (under review)*.
- Zhenghua Chen, **Qingchang Zhu**, Mustafa K. Masood, Yeng Chai Soh, “Environment Sensors based Occupancy Estimation in Buildings via IHMM-MLR,” *IEEE Transactions on Industrial Informatics 13.5 (2017): 2184-2193*.
- Zhenghua Chen, Rui Zhao, **Qingchang Zhu**, Mustafa K. Masood, Yeng Chai Soh, Kezhi Mao, “Building Occupancy Estimation with Environmental Sensors via CDBLSTM,” *IEEE Transactions on Industrial Electronics 64.12 (2017): 9549-9559*.

Conference Proceedings

- **Qingchang Zhu**, Zhenghua Chen and Yeng Chai Soh, “Using unlabeled acoustic data with locality-constrained linear coding for energy-related activity recognition in buildings,” In *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*, pp. 174-179. IEEE, 2015.

- **Qingchang Zhu**, Zhenghua Chen and Yeng Chai Soh, “Smartphone-based Human Activity Recognition in buildings using Locality-constrained Linear Coding,” In *Industrial Electronics and Applications (ICIEA), 2015 IEEE 10th Conference on*, pp. 214-219. IEEE, 2015.

Bibliography

- [1] Annual energy outlook 2017 with projection to 2050, us energy information administration. [https://www.eia.gov/outlooks/aeo/pdf/0383\(2017\).pdf](https://www.eia.gov/outlooks/aeo/pdf/0383(2017).pdf). Accessed: 2017-03-14.
- [2] Afrooz Ebadat, Giulio Bottegal, Damiano Varagnolo, Bo Wahlberg, and Karl H Johansson. Regularized deconvolution-based approaches for estimating room occupancies. *IEEE Transactions on Automation Science and Engineering*, 12(4):1157–1168, 2015.
- [3] Fulin Wang, Qingqing Feng, Zheliang Chen, Qianchuan Zhao, Zhijing Cheng, Jianhong Zou, Yufeng Zhang, Jinbo Mai, Yun Li, and Hayden Reeve. Predictive control of indoor environment using occupant number detected by video data and co 2 concentration. *Energy and Buildings*, 2017.
- [4] Catalina Spataru and Stephanie Gauthier. How to monitor people smartly to help reducing energy consumption in buildings? *Architectural Engineering and design management*, 10(1-2):60–78, 2014.
- [5] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, pages 1–6. ACM, 2010.
- [6] Varick L Erickson, Miguel Á Carreira-Perpiñán, and Alberto E Cerpa. Observe: Occupancy-based system for efficient reduction of hvac energy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 258–269. IEEE, 2011.

- [7] Varick L Erickson, Stefan Achleitner, and Alberto E Cerpa. Poem: Power-efficient occupancy-based energy management system. In *Information Processing in Sensor Networks (IPSN), 2013 ACM/IEEE International Conference on*, pages 203–216. IEEE, 2013.
- [8] H Burak Gunay, William O’Brien, and Ian Beausoleil-Morrison. Implementation and comparison of existing occupant behaviour models in energyplus. *Journal of Building Performance Simulation*, 9(6):567–588, 2016.
- [9] Frauke Oldewurtel, David Sturzenegger, and Manfred Morari. Importance of occupancy information for building climate control. *Applied energy*, 101: 521–532, 2013.
- [10] Adil Al-Mumin, Omar Khattab, and Gopiseti Sridhar. Occupants behavior and activity patterns influencing the energy consumption in the kuwaiti residences. *Energy and buildings*, 35(6):549–559, 2003.
- [11] Anthony Michael Fadell, Matthew Lee Rogers, Kipp Avery Rogers, Abraham K Ishihara, Shahar Ben-Menahem, and Rangoli Sharan. Occupancy pattern detection, estimation and prediction, August 13 2013. US Patent 8,510,255.
- [12] Anthony Michael Fadell, Matthew Lee Rogers, Kipp Avery Rogers, Abraham K Ishihara, Shahar Ben-Menahem, and Rangoli Sharan. Occupancy pattern detection, estimation and prediction, December 16 2015. US Patent App. 14/971,799.
- [13] Tuan Anh Nguyen and Marco Aiello. Energy intelligent buildings based on user activity: A survey. *Energy and buildings*, 56:244–257, 2013.
- [14] Ching-Hu Lu, Chao-Lin Wu, Tsung-Han Yang, Hui-Wen Yeh, Mao-Yung Weng, Li-Chen Fu, and Tsung-Yuan Charlie Tai. Energy-responsive aggregate context for energy saving in a multi-resident environment. *IEEE transactions on automation science and engineering*, 11(3):715–729, 2014.
- [15] Frank Englert, Irina Diaconita, Andreas Reinhardt, Alaa Alhamoud, Richard Meister, Lucas Backert, and Ralf Steinmetz. Reduce the number of sensors: Sensing acoustic emissions to estimate appliance energy usage. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–8. ACM, 2013.

- [16] Hamed Nabizadeh Rafsanjani, Changbum R Ahn, and Mahmoud Alahmad. A review of approaches for sensing, understanding, and improving occupancy-related energy-use behaviors in commercial buildings. *Energies*, 8(10):10996–11029, 2015.
- [17] Ryan Melfi, Ben Rosenblum, Bruce Nordman, and Ken Christensen. Measuring building occupancy using existing network infrastructure. In *Green Computing Conference and Workshops (IGCC), 2011 International*, pages 1–8. IEEE, 2011.
- [18] Thiago Teixeira, Gershon Dublon, and Andreas Savvides. A survey of human-sensing: Methods for detecting presence, count, location, track, and identity. *ACM Computing Surveys (CSUR)*, 5(1), 2010.
- [19] Alessandra De Paola, Marco Ortolani, Giuseppe Lo Re, Giuseppe Anastasi, and Sajal K Das. Intelligent management systems for energy efficiency in buildings: A survey. *ACM Computing Surveys (CSUR)*, 47(1):13, 2014.
- [20] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *IEEE Communications magazine*, 48(9), 2010.
- [21] Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neuro-computing*, 6(2):163–180, 1994.
- [22] Le Zhang and Ponnuthurai N Suganthan. A comprehensive evaluation of random vector functional link networks. *Information sciences*, 367:1094–1105, 2016.
- [23] Le Zhang and Ponnuthurai Nagaratnam Suganthan. Benchmarking ensemble classifiers with novel co-trained kernel ridge regression and random vector functional link ensembles [research frontier]. *IEEE Computational Intelligence Magazine*, 12(4):61–72, 2017.
- [24] Le Zhang and Ponnuthurai N Suganthan. A survey of randomized algorithms for training neural networks. *Information Sciences*, 364:146–155, 2016.
- [25] Ye Ren, Ponnuthurai N Suganthan, Narasimalu Srikanth, and Gehan Amaratunga. Random vector functional link network for short-term electricity load demand forecasting. *Information Sciences*, 367:1078–1093, 2016.

- [26] Le Zhang and Ponnuthurai Nagaratnam Suganthan. Visual tracking with convolutional random vector functional link network. *IEEE transactions on cybernetics*, 47(10):3243–3253, 2017.
- [27] Jessen Page, Darren Robinson, Nicolas Morel, and J-L Scartezzini. A generalised stochastic model for the simulation of occupant presence. *Energy and buildings*, 40(2):83–98, 2008.
- [28] Chenda Liao, Yashen Lin, and Prabir Barooah. Agent-based and graphical modelling of building occupancy. *Journal of Building Performance Simulation*, 5(1):5–25, 2012.
- [29] Chuang Wang, Da Yan, and Yi Jiang. A novel approach for building occupancy simulation. In *Building simulation*, volume 4, pages 149–167. Springer, 2011.
- [30] Zhenghua Chen and Yeng Chai Soh. Modeling building occupancy using a novel inhomogeneous markov chain approach. In *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 1079–1084. IEEE, 2014.
- [31] Zhenghua Chen, Jinming Xu, and Yeng Chai Soh. Modeling regular occupancy in commercial buildings using stochastic models. *Energy and Buildings*, 103:216–223, 2015.
- [32] Varick L Erickson, Yiqing Lin, Ankur Kamthe, Rohini Brahme, Amit Surana, Alberto E Cerpa, Michael D Sohn, and Satish Narayanan. Energy efficient building environment control strategies using real-time occupancy measurements. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 19–24. ACM, 2009.
- [33] Yoon Soo Lee and Ali M Malkawi. Simulating multiple occupant behaviors in buildings: an agent-based modeling approach. *Energy and Buildings*, 69: 407–416, 2014.
- [34] Zhengwei Li, Yeonsook Heo, and Godfried Augenbroe. Hvac design informed by organizational simulation. In *Proceedings of IBPSA Conference*, 2009.
- [35] Elie Azar and Carol C Menassa. Agent-based modeling of occupants and their impact on energy use in commercial buildings. *Journal of Computing in Civil Engineering*, 26(4):506–518, 2011.

- [36] Zhenghua Chen and Yeng Chai Soh. Comparing occupancy models and data mining approaches for regular occupancy prediction in commercial buildings. *Journal of Building Performance Simulation*, pages 1–9, 2016.
- [37] Pengcheng Liu, Sing-Kiong Nguang, and Ashton Partridge. Occupancy inference using pyroelectric infrared sensors through hidden markov models. *IEEE Sensors Journal*, 16(4):1062–1068, 2016.
- [38] Nan Li, Gulben Calis, and Burcin Becerik-Gerber. Measuring and monitoring occupancy with an rfid based system for demand-driven hvac operations. *Automation in construction*, 24:89–99, 2012.
- [39] Sebastian Uziel, Thomas Elste, Wolfram Kattaneck, Danilo Hollosi, Stephan Gerlach, and Stefan Goetze. Networked embedded acoustic processing system for smart building applications. In *Design and Architectures for Signal and Image Processing (DASIP), 2013 Conference on*, pages 349–350. IEEE, 2013.
- [40] Yang Zhao, Wim Zeiler, Gert Boxem, and Timi Labeodan. Virtual occupancy sensors for real-time occupancy information in buildings. *Building and Environment*, 93:9–20, 2015.
- [41] Ming Jin, Ruoxi Jia, Zhaoyi Kang, Ioannis C Konstantakopoulos, and Costas J Spanos. Presencesense: zero-training algorithm for individual presence detection based on power monitoring. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 1–10. ACM, 2014.
- [42] Ming Jin, Ruoxi Jia, and Costas Spanos. Virtual occupancy sensing: Using smart meters to indicate your presence. *IEEE Transactions on Mobile Computing*, 2017.
- [43] Jie Zhao, Bertrand Lasternas, Khee Poh Lam, Ray Yun, and Vivian Loftness. Occupant behavior and schedule modeling for building energy simulation through office appliance power consumption data mining. *Energy and Buildings*, 82:341–355, 2014.
- [44] Saandeep Depatla, Arjun Muralidharan, and Yasamin Mostofi. Occupancy estimation using only wifi power measurements. *IEEE Journal on Selected Areas in Communications*, 33(7):1381–1393, 2015.

- [45] Haochao Li, Eddie CL Chan, Xiaonan Guo, Jiang Xiao, Kaishun Wu, and Lionel M Ni. Wi-counter: smartphone-based people counter using crowd-sourced wi-fi signal data. *IEEE Transactions on Human-Machine Systems*, 45(4):442–452, 2015.
- [46] Ming Jin, Nikolaos Bekiaris-Liberis, Kevin Weekly, Costas J Spanos, and Alexandre M Bayen. Occupancy detection via environmental sensing. *IEEE Transactions on Automation Science and Engineering*, 2016.
- [47] Chaoyang Jiang, Mustafa K Masood, Yeng Chai Soh, and Hua Li. Indoor occupancy estimation from carbon dioxide concentration. *Energy and Buildings*, 131:132–141, 2016.
- [48] Mustafa K Masood, Yeng Chai Soh, and Victor W-C Chang. Real-time occupancy estimation using environmental parameters. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.
- [49] Bing Dong and Burton Andrews. Sensor-based occupancy behavioral pattern recognition for energy and comfort management in intelligent buildings. In *Proceedings of building simulation*, pages 1444–1451, 2009.
- [50] Bing Dong, Burton Andrews, Khee Poh Lam, Michael Höynck, Rui Zhang, Yun-Shang Chiou, and Diego Benitez. An information technology enabled sustainability test-bed (itest) for occupancy detection through an environmental sensing network. *Energy and Buildings*, 42(7):1038–1046, 2010.
- [51] Stylianos Zikos, Apostolos Tsolakis, Dimitrios Meskos, Athanasios Tryferidis, and Dimitrios Tzovaras. Conditional random fields-based approach for real-time building occupancy estimation with multi-sensory networks. *Automation in Construction*, 68:128–145, 2016.
- [52] Junjing Yang, Mattheos Santamouris, and Siew Eang Lee. Review of occupancy sensing systems and occupancy modeling methodologies for the application in institutional buildings. *Energy and Buildings*, 121:344–349, 2016.
- [53] Zheng Yang, Nan Li, Burcin Becerik-Gerber, and Michael Orosz. A systematic approach to occupancy modeling in ambient sensor-rich buildings. *Simulation*, 90(8):960–977, 2014.

- [54] Luis M Candanedo and Véronique Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and co 2 measurements using statistical learning models. *Energy and Buildings*, 112:28–39, 2016.
- [55] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
- [56] 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, 2014.
- [57] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *Pervasive computing*, pages 1–17. Springer, 2004.
- [58] Jamie A Ward, Paul Lukowicz, Gerhard Troster, and Thad E Starner. Activity recognition of assembly tasks using body-worn microphones and accelerometers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1553–1567, 2006.
- [59] Mi Zhang and Alexander A Sawchuk. Human daily activity recognition with sparse representation using wearable sensors. *Biomedical and Health Informatics, IEEE Journal of*, 17(3):553–560, 2013.
- [60] Sourav Bhattacharya, Petteri Nurmi, Nils Hammerla, and Thomas Plötz. Using unlabeled data in a sparse-coding framework for human activity recognition. *Pervasive and Mobile Computing*, 15:242–262, 2014.
- [61] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN*, 2013.
- [62] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczech, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkel, Alois Ferscha, et al. Collecting complex activity datasets in highly rich networked sensor environments. In *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pages 233–240. IEEE, 2010.
- [63] Mi Zhang and Alexander A Sawchuk. Usc-had: a daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the*

- 2012 *ACM Conference on Ubiquitous Computing*, pages 1036–1043. ACM, 2012.
- [64] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul JM Havinga. A survey of online activity recognition using mobile phones. *Sensors*, 15(1):2059–2085, 2015.
- [65] Jafet Morales, David Akopian, and Sos Agaian. Human activity recognition by smartphones regardless of device orientation. In *IS&T/SPIE Electronic Imaging*, pages 90300I–90300I. International Society for Optics and Photonics, 2014.
- [66] Apiwat Henpraserttae, Surapa Thiemjarus, and Sanparith Marukatat. Accurate activity recognition using a mobile phone regardless of device orientation and location. In *Body Sensor Networks (BSN), 2011 International Conference on*, pages 41–46. IEEE, 2011.
- [67] Nicholas D Lane, Mashfiqui Mohammad, Mu Lin, Xiaochao Yang, Hong Lu, Shahid Ali, Afsaneh Doryab, Ethan Berke, Tanzeem Choudhury, and Andrew Campbell. Bewell: A smartphone application to monitor, model and promote wellbeing. In *5th international ICST conference on pervasive computing technologies for healthcare*, pages 23–26, 2011.
- [68] Tae-Seong Kim, Jin-Ho Cho, and Jeong Tai Kim. Mobile motion sensor-based human activity recognition and energy expenditure estimation in building environments. In *Sustainability in Energy and Buildings*, pages 987–993. Springer, 2013.
- [69] Ulf Blanke and Bernt Schiele. Remember and transfer what you have learned—recognizing composite activities based on activity spotting. In *Wearable Computers (ISWC), 2010 International Symposium on*, pages 1–8. IEEE, 2010.
- [70] Mi Zhang and Alexander A Sawchuk. A feature selection-based framework for human activity recognition using wearable multimodal sensors. In *Proceedings of the 6th International Conference on Body Area Networks*, pages 92–98. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011.

- [71] Christian Vollmer, Horst-Michael Gross, and Julian P Eggert. Learning features for activity recognition with shift-invariant sparse coding. In *International Conference on Artificial Neural Networks*, pages 367–374. Springer, 2013.
- [72] Thomas Plötz, Nils Y Hammerla, and Patrick Olivier. Feature learning for activity recognition in ubiquitous computing. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1729, 2011.
- [73] Ming Zeng, Le T Nguyen, Bo Yu, Ole J Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*, pages 197–205. IEEE, 2014.
- [74] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.
- [75] Piero Zappi, Clemens Lombriser, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini, and Gerhard Tröster. Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection. In *Wireless sensor networks*, pages 17–33. Springer, 2008.
- [76] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *IJCAI*, pages 3995–4001, 2015.
- [77] Mohammad Abu Alsheikh, Ahmed Selim, Dusit Niyato, Linda Doyle, Shaowei Lin, and Hwee-Pink Tan. Deep activity recognition models with triaxial accelerometers. *arXiv preprint arXiv:1511.04664*, 2015.
- [78] Maja Stikic, Diane Larlus, Sandra Ebert, and Bernt Schiele. Weakly supervised recognition of daily life activities with wearable sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2521–2537, 2011.
- [79] Xiaojin Zhu. Semi-supervised learning. In *Encyclopedia of Machine Learning*, pages 892–897. Springer, 2011.

- [80] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.
- [81] Maja Stikic, Kristof Van Laerhoven, and Bernt Schiele. Exploring semi-supervised and active learning for activity recognition. In *Wearable computers, 2008. ISWC 2008. 12th IEEE international symposium on*, pages 81–88. IEEE, 2008.
- [82] Maja Stikic, Diane Larlus, and Bernt Schiele. Multi-graph based semi-supervised learning for activity recognition. In *Wearable Computers, 2009. ISWC'09. International Symposium on*, pages 85–92. IEEE, 2009.
- [83] Xinze Guan, Raviv Raich, and Weng-Keen Wong. Efficient multi-instance learning for activity recognition from time series data using an auto-regressive hidden markov model. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2330–2339, 2016.
- [84] 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 *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 13–24. ACM, 2016.
- [85] Seyed Ali Rokni and Hassan Ghasemzadeh. Synchronous dynamic view learning: A framework for autonomous training of activity recognition models using wearable sensors. In *Information processing and sensor network. IPSN'17. International Conference on*. IEEE, 2017.
- [86] Richard F Lyon. Machine hearing: An emerging field [exploratory dsp]. *Ieee signal processing magazine*, 27(5):131–139, 2010.
- [87] Dan Stowell and Mark D Plumbley. Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. *PeerJ*, 2: e488, 2014.
- [88] Jonathan William Dennis. *Sound event recognition in unstructured environments using spectrogram image processing*. PhD thesis, Ph. D. dissertation, Nanyang Technological University, Singapore, 2014.

- [89] Andrey Temko, Robert Malkin, Christian Zieger, Dušan Macho, Climent Nadeu, and Maurizio Omologo. Clear evaluation of acoustic event detection and classification systems. In *International Evaluation Workshop on Classification of Events, Activities and Relationships*, pages 311–322. Springer, 2006.
- [90] Dan Stowell, Dimitrios Giannoulis, Emmanouil Benetos, Mathieu Lagrange, and Mark D Plumbley. Detection and classification of acoustic scenes and events. *IEEE Transactions on Multimedia*, 17(10):1733–1746, 2015.
- [91] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Tut database for acoustic scene classification and sound event detection. In *24th European Signal Processing Conference*, 2016.
- [92] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *IEEE ICASSP*, 2017.
- [93] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.
- [94] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- [95] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366, 1980.
- [96] Hong Lu, Wei Pan, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 165–178. ACM, 2009.

- [97] Hong Lu, AJ Bernheim Brush, Bodhi Priyantha, Amy K Karlson, and Jie Liu. Speakersense: Energy efficient unobtrusive speaker identification on mobile phones. In *International Conference on Pervasive Computing*, pages 188–205. Springer, 2011.
- [98] Robert Anniés, E Martinez Hernandez, Kamil Adiloglu, Hendrik Purwins, and Klaus Obermayer. Classification schemes for step sounds based on gammatone-filters. In *NIPS-Workshop Music, Brain, & Cognition. Citeseer*. Citeseer, 2007.
- [99] Yang Shao, Zhaozhang Jin, DeLiang Wang, and Soundararajan Srinivasan. An auditory-based feature for robust speech recognition. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 4625–4628. IEEE, 2009.
- [100] Alain Rakotomamonjy and Gilles Gasso. Histogram of gradients of time-frequency representations for audio scene classification. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(1):142–153, 2015.
- [101] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.
- [102] Thomas C Walters. *Auditory-based processing of communication sounds*. PhD thesis, University of Cambridge, 2011.
- [103] Jonathan Dennis, Huy Dat Tran, and Haizhou Li. Spectrogram image feature for sound event classification in mismatched conditions. *Signal Processing Letters, IEEE*, 18(2):130–133, 2011.
- [104] Jonathan Dennis, Huy Dat Tran, and Eng Siong Chng. Overlapping sound event recognition using local spectrogram features and the generalised hough transform. *Pattern Recognition Letters*, 34(9):1085–1093, 2013.
- [105] Selina Chu, Shrikanth Narayanan, and CC Jay Kuo. Environmental sound recognition with time–frequency audio features. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(6):1142–1158, 2009.

- [106] Toni Heittola, Annamaria Mesaros, Tuomas Virtanen, and Antti Eronen. Sound event detection in multisource environments using source separation. *Proc CHiME*, pages 36–40, 2011.
- [107] Mingming Zhang, Weifeng Li, Longbiao Wang, Jianguo Wei, Zhiyong Wu, and Qingmin Liao. Sparse coding for sound event classification. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*, pages 1–5. IEEE, 2013.
- [108] J. Salamon and J. P. Bello. Unsupervised feature learning for urban sound classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 International Conference on*. IEEE, 2015.
- [109] Lars Hertel, Huy Phan, and Alfred Mertins. Comparing time and frequency domain for audio event recognition using deep learning. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 3407–3411. IEEE, 2016.
- [110] Erik Marchi, Fabio Vesperini, Florian Eyben, Stefano Squartini, and Björn Schuller. A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional lstm neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 1996–2000. IEEE, 2015.
- [111] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Recurrent neural networks for polyphonic sound event detection in real life recordings. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 6440–6444. IEEE, 2016.
- [112] Ian McLoughlin, Haomin Zhang, Zhipeng Xie, Yan Song, and Wei Xiao. Robust sound event classification using deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):540–552, 2015.
- [113] Jonathan Dennis, Huy Dat Tran, and Haizhou Li. Combining robust spike coding with spiking neural networks for sound event classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 176–180. IEEE, 2015.

- [114] Sumit Bam Shrestha. *Supervised learning in multilayer spike neural network*. PhD thesis, Ph. D. dissertation, Nanyang Technological University, Singapore, 2017.
- [115] Varick L Erickson, Miguel Á Carreira-Perpiñán, and Alberto E Cerpa. Occupancy modeling and prediction for building energy management. *ACM Transactions on Sensor Networks (TOSN)*, 10(3):42, 2014.
- [116] Jiakang Lu, Tamim Sookoor, Vijay Srinivasan, Ge Gao, Brian Holben, John Stankovic, Eric Field, and Kamin Whitehouse. The smart thermostat: using occupancy sensors to save energy in homes. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 211–224. ACM, 2010.
- [117] Guang-Bin Huang, Zuo Bai, Liyanaarachchi Lekamalage Chamara Kasun, and Chi Man Vong. Local receptive fields based extreme learning machine. *IEEE Computational Intelligence Magazine*, 10(2):18–29, 2015.
- [118] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- [119] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [120] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
- [121] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.
- [122] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [123] Wouter F Schmidt, Martin A Kraaijveld, and Robert PW Duin. Feedforward neural networks with random weights. In *Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, pages 1–4. IEEE, 1992.

- [124] Yoh Han Pao and Y Takefji. Functional-link net computing. *IEEE Computer Journal*, 25(5):76–79, 1992.
- [125] Boris Igel'nik and Yoh-Han Pao. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks*, 6(6):1320–1329, 1995.
- [126] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 985–990. IEEE, 2004.
- [127] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.
- [128] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529, 2012.
- [129] Giuseppe Parise, Luigi Martirano, and Giorgio Cecchini. Design and energetic analysis of an advanced control upgrading existing lighting systems. *IEEE Transactions on Industry Applications*, 50(2):1338–1347, 2014.
- [130] Zhenghua Chen, Rui Zhao, Qingchang Zhu, Mustafa K Masood, Yeng C Soh, and Kezhi Mao. Building occupancy estimation with environmental sensors via cdblstm. *IEEE Transactions on Industrial Electronics*, 2017.
- [131] Najdan Vukovic, Milica Petrovic, and Zoran Miljkovic. A comprehensive experimental evaluation of orthogonal polynomial expanded random vector functional link neural networks for regression. *Applied Soft Computing*, 2017.
- [132] Pablo A Henriquez and Gonzalo A Ruz. A non-iterative method for pruning hidden neurons in neural networks with random weights. *Applied Soft Computing*, 2018.
- [133] Ling Tang, Yao Wu, and Lean Yu. A non-iterative decomposition-ensemble learning paradigm using rvfl network for crude oil price forecasting. *Applied Soft Computing*, 2017.

- [134] Diego Parente Paiva Mesquita, Joao Paulo Pordeus Gomes, Leonardo Ramos Rodrigues, Saulo A F Oliveira, and Roberto Kawakami Harrop Galvao. Building selective ensembles of randomization based neural networks with the successive projections algorithm. *Applied Soft Computing*, 2017.
- [135] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.
- [136] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [137] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2009.
- [138] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [139] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. Data augmentation for time series classification using convolutional neural networks. In *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, 2016.
- [140] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [141] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1163–1171, 2016.
- [142] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.
- [143] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

- [144] Sander Dieleman, Jan Schlter, Colin Raffel, Eben Olson, Sren Kaae Snderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Diogo Moitinho de Almeida, Brian McFee, Hendrik Weideman, Gbor Takcs, Peter de Rivaz, Jon Crall, Gregory Sanders, Kashif Rasul, Cong Liu, Geoffrey French, and Jonas Degraev. Lasagne: First release., August 2015. URL <http://dx.doi.org/10.5281/zenodo.27878>.
- [145] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.
- [146] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [147] Ian Cleland, Basel Kikhia, Chris Nugent, Andrey Boytsov, Josef Hallberg, Kåre Synnes, Sally McClean, and Dewar Finlay. Optimal placement of accelerometers for the detection of everyday activities. *Sensors*, 13(7):9183–9200, 2013.
- [148] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360–3367. IEEE, 2010.
- [149] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.
- [150] Max Welling. Fisher linear discriminant analysis. *Department of Computer Science, University of Toronto*, 3, 2005.
- [151] Senjian An, Wanquan Liu, and Svetha Venkatesh. Face recognition using kernel ridge regression. In *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*, pages 1–7. IEEE, 2007.
- [152] Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In *Machine Learning (ICML), 15th IEEE International Conference on*. IEEE, 1998.

- [153] Guang-Bin Huang. An insight into extreme learning machines: random neurons, random features and kernels. *Cognitive Computation*, 6(3):376–390, 2014.
- [154] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
- [155] Zhenghua Chen, Qingchang Zhu, Chai Soh Yeng, and Le Zhang. Robust human activity recognition using smartphone sensors via ct-pca and online svm. *IEEE Transactions on Industrial Informatics*, 2017.
- [156] Daniel P. W. Ellis. PLP and RASTA (and MFCC, and inversion) in Matlab, 2005. URL <http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/>. online web resource.
- [157] Antti J Eronen, Vesa T Peltonen, Juha T Tuomi, Anssi P Klapuri, Seppo Fagerlund, Timo Sorsa, Gaëtan Lorho, and Jyri Huopaniemi. Audio-based context recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(1):321–329, 2006.
- [158] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
- [159] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018. ACM, 2015.
- [160] Karol J Piczak. Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2015.
- [161] Yaxiong Xie, Zhenjiang Li, and Mo Li. Precise power delay profiling with commodity wifi. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 53–64. ACM, 2015.
- [162] Xuyu Wang, Lingjun Gao, and Shiwen Mao. Csi phase fingerprinting for indoor localization with a deep learning approach. *IEEE Internet of Things Journal*, 3(6):1113–1123, 2016.

-
- [163] Fadel Adib and Dina Katabi. *See through walls with WiFi!*, volume 43. ACM, 2013.
 - [164] Fadel Adib, Hongzi Mao, Zachary Kabelac, Dina Katabi, and Robert C Miller. Smart homes that monitor breathing and heart rate. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pages 837–846. ACM, 2015.
 - [165] Fadel Adib, Zachary Kabelac, and Dina Katabi. Multi-person localization via rf body reflections. In *NSDI*, pages 279–292, 2015.
 - [166] Mingmin Zhao, Fadel Adib, and Dina Katabi. Emotion recognition using wireless signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 95–108. ACM, 2016.