Building Occupancy and Thermal Modelling in the Wild

by

Tianyu Zhang

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Occupancy and thermal modelling is the foundation of several smart building applications, such as intelligent control of residential and commercial buildings' Heating, Ventilation and Air Conditioning (HVAC) system to improve energy efficiency and overall occupant experience in the built environment. And yet developing occupancy and thermal models is quite challenging due to the lack of information about the building's layout and structure, and reliable means for collecting ground truth about occupant presence and actions. Thus, most HVAC systems are presently operated without incorporating these models, a practice that has been proven to be extremely inefficient and inflexible. This thesis aims to address the challenges of building suitable models using data collected by sensors that are normally installed in buildings. We propose black-box models to estimate the occupancy state of the building, and grey-box lumped parameter thermal models to explain how the indoor temperature changes over time. The black-box occupancy models are built using the data acquired through the Building Management System (BMS), while the grey-box thermal models are built using the data reported by smart thermostats. We present a methodology for transferring occupancy estimation models from a controlled environment, where training data is abundant, to another similar environment, where training data is sparse or non-existent. Experiments run using real data collected from a large number of buildings corroborate that the transferred models can achieve high accuracy. Moreover, the thesis presents an open-source toolkit, which enables the development and evaluation of various

occupancy estimation models across different buildings. Several case studies are presented to show the usefulness and extensibility of this toolkit.

Preface

This thesis is original work by Tianyu Zhang. Some of the chapters are based on conference publications co-authored by the author of this thesis.

Specifically, Chapter 3 was originally published as a conference paper: T. Zhang and O. Ardakanian, "A Domain Adaptation Technique for Fine-Grained Occupancy Estimation in Commercial Buildings," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, IoTDI '19, ACM, 2019, pp. 148–159. I was responsible for conducting the literature review, developing the methodology, running experiments, and producing results. The other author is my supervisor who edited the manuscript.

Chapter 4 was also published as a conference paper: T. Zhang, A. Al Zishan, and O. Ardakanian, "ODToolkit: A Toolkit for Building Occupancy Detection," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, ACM, 2019, pp. 35–46. I was responsible for conducting the literature review, developing the methodology (except two models which are mentioned in the thesis), running case studies, and building and documenting the toolkit. A. Al Zishan assisted with the development of the two models mentioned in the thesis and contributed to writing one section of the paper. O. Ardakanian edited the manuscript and designed the case studies.

Acknowledgements

I am greatly indebted to my supervisor, Professor Omid Ardakanian, for giving tons of valuable suggestions during my graduate studies and carefully reviewing my thesis. He walked me through all the stages of writing this thesis and the papers we published before. Without his patient instruction and expert guidance, the completion of this thesis would be impossible.

Special thanks to my friends who have put considerable time and effort into their comments on the draft.

Finally, I must express my gratitude to my beloved girlfriend and family for providing me with unfailing support and continuous encouragement, and my burned graphics card for its life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, smart networked systems have become increasingly popular as they are capable of monitoring their surrounding environment and modelling it dynamics by fusing data from a variety of sensors. Unlike human-in-the-loop control systems, these systems have a higher degree of autonomy, relying mostly on the sensor readings and other (possibly external) sources of information to update their control settings or complete a desired task with minimal human intervention. For example, smart thermostats, which are prevalent in modern homes and buildings, can *sense* the ambient air temperature and occupants' presence, and download the electricity price and weather forecasts from the Internet. This data is used to proactively adjust the temperature setpoint of heating or cooling equipment. The quality of data available to such a system would greatly affect its performance. If the embedded sensors are noisy, or have a low sampling frequency when the environment is highly variable, the system operation would become sub-optimal. In case of a smart thermostat, the system can be "fooled" by noisy estimates of the occupancy state to turn on heating or cooling equipment to condition an empty room.

A smart system can utilize different sensing modalities to monitor its environment. For example, the occupancy state of a room can be inferred from the measurements of a passive infrared (PIR) sensor, a carbon-dioxide (CO_2) sensor, a thermal sensor, or a camera. Some of these sensors offer a more reliable estimate of the occupancy state, often at the cost of being more expensive and intrusive. Other sensors are *zero-cost* as they are commonly installed in

buildings for other purposes, e.g., a carbon-dioxide sensor feeds nested control loops of the Heating Ventilation and Air Conditioning (HVAC) system and its measurements are logged by the Building Management System (BMS).

1.1 The need for building occupancy and thermal modelling

A large fraction of smart systems are currently installed in homes and buildings to ensure safety and security of occupants, improve their thermal comfort, reduce energy consumption of various building subsystems, and support other building applications. Examples of these applications are demand-controlled ventilation, occupancy-driven lighting control, physical security, and space utilization [17], [100], [111], which could reduce the energy consumption of buildings [25] and improve occupant experience and thermal comfort. Furthermore, most control applications require a model to predict building-level or room-level occupancy and a model that explains the heat flow inside the building to make *optimal control* decisions regrading the operation of a building subsystem. In particular, recent studies suggest that incorporating fine-grained occupancy information (i.e., presence and count) in building controls can remarkably improve human experience and operational efficiency of HVAC [1], [5] and lighting systems [79]. According to the data published by U.S. Department of Energy, the HVAC system uses more energy than any other building subsystem, which is around 47% of the total energy consumption for residential buildings [36] and 43% of the total energy consumption for commercial buildings [40]. Thus, it is imperative to improve the efficiency of the HVAC system, which highlights the significance of modelling occupancy and temperature inside the building.

Modelling building occupancy and temperature is a challenging task. This is mainly due to (a) the lack of special-purpose sensors for occupancy monitoring, such as cameras, which are costly and carry a heavy deployment stigma, (b) the highly uncertain and complex nature of occupancy and temperature dynamics in the built environment, and (c) insufficient labelled data. Existing

approaches to incorporate occupancy in the control loops require retrofitting the building with numerous occupancy sensors, such as cameras, microphones, PIR, CO_2, and illumination sensors, which is intrusive, costly, and error-prone given the scale.

There are multiple approaches to determine the real-time occupant presence and count in the built environment. The first approach builds high-dimensional heat transfer models, such as the lumped parameter thermal model [99] (aka Resistance-Capacitance (RC) network model), and uses them to infer the internal heat gain in individual rooms. This heat gain is attributed to room-level occupancy [96], [108] subsequently. The RC models describe temperature dynamics inside the building according to heat transfer and thermodynamics theories, but they must be customized for each room based on its size, layout, and *envelope*[1]. Thus, the RC models cannot be simply used to distinguish the effect of occupancy from other confounding effects. The second approach builds data-driven models to determine occupant presence and count in the many rooms in a building [5], [9], [10], [58]. These models are easier to build and can substitute complex physics-based models with an insignificant loss of prediction accuracy [107]. However, developing such models requires extensive training with a significant amount of labelled occupancy data, which is not ordinarily available for different rooms and buildings. The third approach uses data from other sources such as work schedules and room booking information for an office or educational building, and household demands for residential buildings to approximate the occupants' presence and count. These estimations generally have low accuracy and the corresponding data feeds are not readily available in all types of buildings. The shortcomings of the first and third approaches motivate the development of data-driven models for occupancy estimation.

Besides the need for estimating and predicting occupancy, another key element that contributes to human comfort is the temperature inside a room. There are several approaches to build thermal models for buildings, including

[1]The building envelope is the physical separator between the building space and outside environment; it includes doors, windows, exterior walls and insulation.

first-principle (white-box), data-driven (black-box), and hybrid (grey-box) approaches. First-principle approaches estimate the indoor temperature based on the physical and mathematical models of the building. The RC model is an example of grey-box model which models heat flow by creating an analog circuit model based on semi-physical laws [61]. Data-driven approaches use various mathematical methods to estimate the trend of the temperature base on real measured historical data. One well-known black-box model for time-series analysis is the seasonal autoregressive integrated moving average with explanatory variable (SARIMAX). This thesis focuses on the development of grey-box and black-box thermal models.

1.2 Challenges

Residential and commercial buildings are complex cyber-physical systems with many subsystems and various dynamics that span multiple time scales; these dynamics cannot be easily modelled for several reasons. For example, to model temperature dynamics of a building, it is necessary to know physical parameters that depend on its size, layout, envelope and structure. Many of these parameters are not typically known for buildings that are in operation today. While it is possible to acquire this knowledge through energy audits, they are time-consuming and expensive. This motivates the use of appropriate analytics to estimate these parameters from sensor data. But dedicated and high-accuracy sensors are not currently installed in most residential and commercial buildings and we have to rely on the data collected by sensors that are commonly available in these buildings to develop portable building applications. Examples of the sensors that are commonly available are smart thermostats and HVAC sensors which we explore in the thesis.

Another critical challenge is that training data is often unavailable or insufficient for occupancy and thermal modelling. In particular, developing data-driven models usually requires a substantial amount of labelled data collected over an extended period of time. This is because occupancy schedules, thermal comfort requirements, and the usage of heating and cooling systems vary

from one season to another in a building. However, reliable ground truth data collection is overly costly and intrusive in real world scenarios since it needs an enormous effort to manually log occupancy events in all rooms for a least several weeks or requires the deployment of multiple cameras in different areas of the building. Thus, to achieve scalable deployment of building applications across the building stock, it is necessary to adapt and reuse existing models that are originally trained for a small set of fully-instrumented test buildings to estimate occupancy and temperature dynamics in other buildings. This can be achieved by applying *transfer learning*.

Transfer learning concerns mapping multiple data sets (from different domains) into a latent space such that they have a similar distribution. In other words, it transfers knowledge from one learned task to a new task, thus improving the learning outcome in the new task [83]. Many recent studies show that transfer learning is quite effective in different areas, such as computer vision [44] and natural language processing [57]. Transfer learning can also help to reduce the amount of data needed to build an accurate model for estimating occupancy and temperature dynamics inside a building [106], because it allows for reusing a well-suited model that is originally built for a test building to another similar building which may not be instrumented.

Finally, recent studies on occupancy modelling utilize different sensing modalities and are typically validated on a small number of test buildings, which do not necessarily represent the occupancy pattern of real buildings. Moreover, there is no consensus on how to report the accuracy of models, especially when models estimate the number of occupants. For these reasons, it is virtually impossible today to evaluate various occupancy detection algorithms and form a conclusive opinion about which sensing modalities are more indicative of occupancy, what sensor precision is necessary, or how much training data (occupancy labels) must be collected to build an accurate model. This manifests the need for a toolkit for occupancy estimation tasks (similar to NILMTK [16] which was designed for non-intrusive load monitoring) that enables researchers to evaluate the performance of their occupancy detection algorithms using a standard set of metrics on data sets collected from multiple

types of buildings in different climates.

1.3 Objectives and contributions

To address the above mentioned challenges, we identify and set the following objectives:

1. Developing accurate data-driven models for estimating the presence and count of occupants inside the many rooms of a building. These models must be trained using data collected by sensors that are commonly available in commercial buildings rather than dedicated sensors for occupancy monitoring which are not deployed in legacy commercial buildings.

2. Developing a toolkit that offers open-source implementation of state-of-the-art occupancy estimation models and contains publicly available data sets pertaining to residential and commercial buildings in several countries around the world. With this toolkit, it will be possible to compare and analyze different occupancy estimation algorithms by adopting standard evaluation metrics.

3. Building black-box and grey-box thermal models for describing how the temperature inside a building responds to changes in the outside temperature and the operation of heating or cooling equipment. To determine the superior model, these two kinds of models are compared in residential buildings without assuming the knowledge of building insulation and thermal mass.

4. Showing empirically how transfer learning can be applied to customize and reuse pre-trained occupancy estimation models in test commercial buildings.

To achieve these objectives we use real data from two commercial buildings, in Canada and Denmark, and data collected by ecobee thermostats from thousands of homes and buildings in Canada.

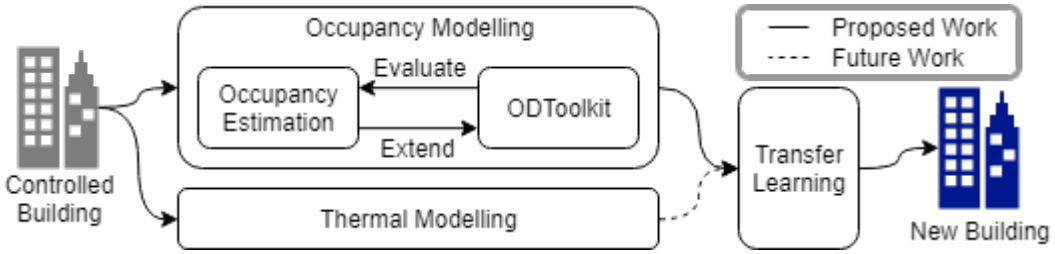


Figure 1.1: A high-level overview of the ideas presented in the thesis.

The contribution of this thesis is three-fold. First, we train long short term memory (LSTM) networks and non-linear autoregressive network with exogenous inputs (NARX) networks for the occupancy estimation task using the HVAC sensor data available through a rudimentary BMS [106]. We demonstrate how these models can be transferred from one zone to another zone located in the same commercial building. Second, we design and implement an occupancy detection toolkit, called ODToolkit [105], hoping that it supports the effort to benchmark new models and algorithms against previous work, and contributes to a subtask of Annex 79 of International Energy Agency (IEA). Third, we describe mathematical derivations for building a low-order RC-network model and explain how a Bayesian neural network (BNN) helps estimate the parameters of this model. The accuracy of this model is predicting the indoor temperature is compared with a black-box time-series model, namely the SARIMAX model. Figure 1.1 summarizes the contribution of this thesis.

1.4 Outline of the thesis

The rest of this thesis is organized as follows. Chapter 2 describes related work on building occupancy and thermal modelling, Bayesian neural network, and different applications of transfer learning. Chapter 3 explains how to build and evaluate occupancy estimation models using data from HVAC sensors, and how to adapt these models so that they can be used to estimation the occupancy of another room with the same commercial building. Chapter 4 focuses on the design and implementation of an occupancy detection toolkit,

and discusses how it can be extended to incorporate new data sets, algorithms, and metrics. It also presents several case studies for evaluating the toolkit. Chapter 5 explains how to build and evaluate thermal models for residential buildings equipped with smart thermostats. Finally, the thesis is concluded in Chapter 6 by specifying the limitations of this work and providing directions for future work.

Chapter 2

Literature Review

This chapter reviews the fundamental concepts related to modelling and control of smart buildings and surveys related work on building occupancy detection, thermal modelling, and transfer learning. It also summarizes recent efforts to build a toolkit for thorough comparison of different algorithms. The rest of this chapter is outlined as follows: Section 2.1 describes occupancy modelling and algorithms proposed in the literature. Section 2.2 provides a detailed overview of thermal modelling, followed by a discussion of transfer learning in Section 2.3. Section 2.4 surveys the related work on building a toolkit.

2.1 Occupancy modelling

Occupancy is one of the main factors determining the building energy consumption. Despite huge variation in occupancy over space and time, the whole building is often treated as a uniform environment controlled with a fixed ventilation rate and static schedules for air conditioning and lighting, thereby wasting much energy in conditioning and lighting empty or partially occupied spaces. Thus, controlling HVAC and lighting systems based on occupancy can result in substantial energy savings and tangible improvements in occupant comfort.

Fine-grained occupancy information is one of the building blocks of many smart building applications, such as the energy-efficient operation of the HVAC system, smart lighting, workspace utilization, security, etc. While most exist-

ing building applications rely on the information about the binary occupancy state (i.e., occupied or empty) of each room, the ability to discern the number of occupants in each room enables even more applications, for example smart lighting, demand-controlled filtration and ventilation [100], space utilization, safety, and evacuation [64]. The more accurately the number of occupants is estimated, the better it would be for the target application. Given the importance of obtaining high-accuracy occupancy information, several models have been developed for occupancy detection in the literature.

2.1.1 Sensing modalities

Several attempts have been made to date to monitor building occupancy using wired and wireless sensor networks. For example, motion sensor data is fused with data from magnetic reed switches [1], thermal sensor arrays [17], carbon-dioxide sensors [3], [7], [8], [15], [60], [96], cameras [37], passive infrared [88], [97], and other ambient sensors [73] to estimate the room-level occupancy state (viz. occupant presence or count). In recent work, a custom sensor tag has been designed which integrates and fuses a large number of sensing modalities [67]. This general-purpose sensor enables monitoring occupancy along with several other environmental facets. Nevertheless, all these systems require hardware retrofits, posing many challenges from sensor placement and calibration to ensuring that the sensors have a reliable network connection and power supply. Moreover, some of them utilize sensors that are recognized for being privacy-invasive (e.g., cameras and microphones) and carry a heavy deployment stigma.

To preserve privacy while maintaining the overall occupancy estimation accuracy, Kjaergaard et al. [64] installed high-precision, dedicated people counting sensors in common areas of a building to count the occupants. Combining these estimates with measurements of low-accuracy sensors, they *disaggregated* the building occupant count into room counts, i.e., they estimated what fraction of building occupants are in each individual room. This solution improves the accuracy of occupancy estimation in the whole building, but requires retrofits to install people counting sensors and a model to normalize the

data collected from each sensor.

To address these limitations, a growing body of research focuses on the opportunistic use of existing building infrastructure to monitor occupancy, when possible. For example, the wireless networking infrastructure is leveraged in [12], [29], [94], [102], [111] to estimate the occupancy state of different spaces in a building. Wireless network includes radio-based devices that use Bluetooth, WiFi, or electromagnetic waves. Using multipath fading (MP) with a line of sight based on received signal strength indication (RSSI), Reference [33] achieves around 96% accuracy for outdoor occupancy. Similarly, building occupancy is inferred in [45] leveraging the wireless networking infrastructure, and security and access control systems. The above approaches assume that occupants always carry WiFi-enabled devices. Even with this assumption, it is often impossible to estimate the occupancy state of each room because an access point may cover multiple rooms.

The majority of medium and large commercial buildings are equipped with HVAC sensors sending their measurements to the central BMS. Therefore, a number of recent studies address the occupancy detection problem by measurements of the room temperature, and damper and valve position, which are parts of the HVAC system, is used to infer the occupancy state of individual rooms [5], [47]. These techniques have two main shortcomings. First, they achieve an acceptable level of accuracy only if physical sensors are installed in suitable locations in the building. Second, modelling occupancy using these techniques requires an abundance of labelled occupancy data collected from the same environment. It is quite challenging to obtain labelled data as it requires substantial manual effort or reliable video-based systems which are costly and intrusive.

Another approach to monitoring building occupancy is to build a high-dimensional physics-based model for heat transfer in a room [59], [99], [107]. The internal heat gain due to occupancy can be inferred from the physical-based model, as shown in [10]. This approach makes strong assumptions about the indoor environment, i.e., the internal heat gain can be attributed to the HVAC system and occupants only and other heat sources, such as appliances,

Table 2.1: Taxonomy of related work on occupancy detection

Category	Sensors	Estimation Type	Method	Granularity	Duration	Ground truth collection	Ref.
New Wired/Wireless Sensor Network	multi-modal sensors	count	Rule-based GP LR	15 min	1 month	tagged w/ app	[73]
	Grid-Eye and PIR sensors	count	KNN ANN LR	0.1 sec	3 weeks	video playback	[17]
	cameras, PIR sensors	count	KNN	0.03 sec	4 weeks	video playback	[37]
	PIR, CO2 sensors	count	ARMA + EM	1.8 sec	1 day	manual logging	[60]
	CO2 sensors	count	SNMF	1 min	13 days	manual logging	[15]
	ambient sensors	count	PF NARX	10 sec	16 days	video playback	[47]
	CO2 sensors	count	Sensing by Proxy Bayes net			manual logging	[59]
	PIR and CO2 sensors	count	DCount	1 min	30 days	video playback	[64]
	ambient sensors camera	count	GAKF CAM	1 min	several months	video playback	[96]
	PIR sensor, reed switch	presence	classification	15 min	4 days	manual logging	[1]
	ambient sensors	presence	RF GBM CART LDA	14 sec	3 weeks	video playback	[23]
	CO2 sensors	count	RUP	5 min	14 days	manual logging	[8]
	CO2 sensors	presence count	DA-HOC++	5 min	2 month	manual logging	[7]
Existing Infrastructure	WiFi	count	Sentinel	15 min	3 weeks	manual logging	[12]
	WiFi	count	LOS + motion	0.02 sec	5 min	manual logging	[33]
	WiFi	count	WinOSS	30 sec	24 weeks	manual logging	[111]
	HVAC sensors	presence	EMD	10 min	3 month	manual logging, camera	[5]
	WiFi, calendar	presence	classification	1 min	6 weeks	tagged w/ app	[45]
	smart meters	presence	NIOM	1 min	7 days	tagged w/ app	[27]

are not present in the indoor environment. This assumption does not hold in practice, hence this approach is not studied further in this thesis.

Table 2.1 shows the taxonomy of related work on occupancy detection using various sensing modalities, algorithms, data sets, and ground truth data collection techniques.

2.1.2 Inference models

An overview of the multitude of data-driven algorithms for occupancy modelling is provided in the following. This thesis divides these data-driven algorithms into two categories, single snapshot prediction models and time-series models. All the algorithms explained below are implemented in ODToolkit as described in Chapter 4.

Single-snapshot models:

Single snapshot models take into account sensor data from the current time slot to estimate the occupancy state of the same time slot. These models do not have a memory; therefore, they ignore data from previous time slots when making predictions about the occupancy state of the present time.
Hidden Markov Model (HMM) is a probabilistic graphical model where the underlying system is assumed to be a Markov process. States are hidden but can be estimated from measurements. HMM is a directed graph with loops, which encodes the possibility of moving from one state to another state and being in a certain state given sensor measurements. The edge weights are transition and emission probabilities learned from the data set. The vertices in this graph are the states and possible output results, i.e., labels. Training a HMM requires a lot of ground truth data. When HMM is used for occupancy detection, the vertices are the number of occupants from 0 to the maximum capability of the room. Measurements of occupancy-indicating sensors are considered as emissions or outputs of the hidden states.

The transition probabilities of HMM are denoted by $P(X_t|X_{t-1})$ where X_t is the occupancy state at time t. If the maximum possible occupancy count is K at time t, then X_t can take one of the values from $0, 1, \ldots, K$.

These probabilities form the transition matrix denoted by $A^{K\times K}$. The emission probabilities are defined by $P(Y_t|X_t)$ where $Y_t = \left[Y_t^{(1)}, Y_t^{(2)}, \ldots, Y_t^{(M)}\right]$ is the measurement vector of M sensors at time t. We assume that sensor measurements are independent and follow Normal distribution:

$$P(Y_t|X_t) = \prod_{i=1}^{M} P(Y_t^{(i)}|X_t)$$
$$P(Y_t|X_t = k) = \mathcal{N}(\hat{\mu}, \hat{\sigma}^2|k)$$

Here, $\hat{\mu}$ and $\hat{\sigma}^2$ are the conditional mean and variance estimated from data. The emission probabilities form a matrix denoted by B:

$$B_i(j) = p(Y_t = j|X_t = i)$$

The initial probability, π, is assumed to be uniform over the hidden states. These three HMM parameters are expressed together as $\theta = (A, B, \pi)$. For training HMMs, this thesis implements the maximum likelihood estimation (MLE) of θ. Specifically, for a sequence of occupancy states (hidden states) and corresponding measurements (emission states), *i.e.*, $\{(X_t, Y_t)\}$, the MLE of the HMM parameters is given by:

$$\theta^* = \arg\max_{\theta} p\left(\{(X_t, Y_t)\}|\theta\right)$$

The optimum parameters, $\theta^* = (A^*, B^*, \pi^*)$ can be expressed as,

$$A_{ij}^* = \frac{\text{\# of transitions from i to j}}{\text{\# of transitions from i}}$$
$$B_i^* = (\mu_i, \sigma_i^2)$$

Here (μ_i, σ_i^2) are respectively the estimated mean and variance of the measurements observed from state i. For prediction, Viterbi algorithm is used to estimate the underlying sequence of hidden states from a given sequence of sensor measurements. This model was used to predict the number of occupants in [2] with an accuracy of over 80%.

Random Forest (RF) leverages a large number of decision trees. The algorithm applies bootstrap aggregating to train the trees. For RF, it selects feature data and the corresponding label randomly (with replacement) to generate a bag,

and train a decision tree on this bag. Each bag is also a randomly selected subset of the features. After generating several such trees, RF can use those trees or the forest to predict the label. For each feature vector, RF applies all trees to the vector and computes the average prediction from each tree to get the final prediction.

This algorithm can maintain a high accuracy even when part of the feature value is missing for the prediction. Also, the algorithm has the ability to compute the importance of each feature. It also runs faster than some other algorithms. Reference [23] shows that the performance of RF in an occupancy estimation task depends on the features. It works best when light and carbon-dioxide data are available, resulting in an accuracy of over 95%.

Support Vector Machine (SVM) is a machine learning algorithm which is utilized to classify data with non-linear boundaries for classification. It finds the hyper-plane that separates most of the data that does not belong the same label. Therefore, the label must be categorical rather than numerical. SVM only works on categorical data, but the occupancy estimation problem is trying to find the function between features and labels. Therefore, we used Support Vector Regression (SVR) to perform occupancy estimation.

SVM has been used in the smart home environment to detect visitors [90], yielding an accuracy of over 67%. It is also used to predict occupancy in [74] from HVAC data with an accuracy of 92%, and in [35] with an accuracy of 82.6%.

Artificial Neural Network (ANN) is a computational approach that mimics the functioning of biological neural networks. ANN is a non-linear statistical data modelling tool and has been very successful for tasks involving pattern recognition. They have proven to be useful in extracting patterns and detecting trends in high-dimensional data sets which cannot be analyzed with other computational models. Within each layer, multiple nodes are acting as neurons. Each node has a relation with all nodes in the next layer. We use weights to present this kind of relation. Also, we add one more node in each layer except the output layer. We use this extra node as the bias node. Biases are always helpful. In effect, a bias value allows the model to shift the activation function

to the left or right, which is critical for successful learning. Reference [35] trains ANN to estimate the occupancy level from HVAC sensors and reports an accuracy of 81.1%.

Particle Filtering (PF) is a well known and suitable approach for solving the state estimation problem in the non-Gaussian and non-linear regime. It is a variant of Sequential Monte Carlo (SMC) where *particles* (or samples) are used to estimate the underlying hidden state sequence. It has been previously used for binary occupancy detection [24], [84] and for occupancy count determination [47], given measurements of multiple sensors.

There are two models associated with PF: the *system model* describing the change of occupancy states over the time and the *measurement model* representing the relation between hidden states and sensor measurements. We implement PF on top of HMM where the system model is equivalent to the transition matrix (i.e., A), and the measurement model is represented by emission probabilities (i.e., B). The only difference between PF and HMM is how they predict the occupancy state.

Algorithm 1 shows different steps of the PF method. It takes a sequence of measurements, $Y_{1:T}$, and gives an estimated sequence of occupancy states, $X_{1:T}$. For each time t, it updates a set of particles based on the previous (i.e. $t-1$) set according to $p(X_t|X_{t-1})$ and assigns some weights. Later the particles are resampled according to these weights. The sample mean of these new particles represents the occupancy state at time t.

Sparse Non-negative Matrix Factorization (SNMF) is a dimensionality reduction algorithm that learns a non-negative low-dimensional representation of sensor data. In *PerCCS* [15], SNMF is used as a pre-processing step to find a low-dimensional representation of CO_2 data. Precisely, it provides reduced noise CO_2 data, which is then fed to an Ensemble Least Square Regression to estimate the occupancy count. PerCSS can estimate the occupancy with a normalized mean squared error (NMSE) of 7.5% and can predict the occupancy count with NMSE of 91% and 15% when the room is unoccupied and occupied, respectively. Consider the implementation of PerCCS where the

Algorithm 1: Particle Filtering

Input: N: number of particles; T: size of the data set
$Y_{1:T}$: measurements up to time T
Output: $\mathcal{X}_{1:T}$: sequence of hidden sates
$\mathcal{X}_{1:T} = \{\}$
for $t = 1$ **to** T **do**
  for $s = 1$ **to** N **do**
    $x_t^{[s]} \sim p(X_t | X_{t-1} = x_{t-1}^{[s]})$
    $w_t^{[s]} = p(Y_t | X_t = x_t^{[s]})$
  end
  Normalize $w_t^{[s]}$
  for $s = 1$ **to** N **do**
    $x_t^{[s]} \sim$ resample according to $w_t^{[s]}$
  end
  $\mathcal{X}_{1:T} = \mathcal{X}_{1:T} + \{average(x_t^{[1:N]})\}$
end

matrix decomposition is:

$$\min_{W \geq 0, H} \frac{1}{2} \|X - WH\|_F^2 + \alpha \|W\|_2^2 + \beta \|W\|_1$$

Here, $X \in \mathbb{R}^{\frac{N}{15} \times 30}$ is the data matrix, $W \in \mathbb{R}^{\frac{N}{15} \times 15}$ is the new representation, N is the number of observation, *time length* (i.e. time required to affect the CO_2 readings) is 30 minutes, *prediction resolution* is 15 minutes, and α and β are the regularization and sparsity coefficient, respectively. must be tuned.

Time-series models

Time-series models utilize several previous data points and current input to predict the current value(s). The structure of these models allows them to memorize the historical input and output data. Recurrent neural networks (RNN) are one of the most popular models to deal with time-series data (i.e., sequences of inputs). In particular, an RNN maintains a state to capture the history of the input sequence, enabling it to learn complex temporal dependencies. In the last few years, there has been an incredible success in applying RNNs to a variety of problems, such as speech recognition [51], language modelling [76], translation [28], etc.

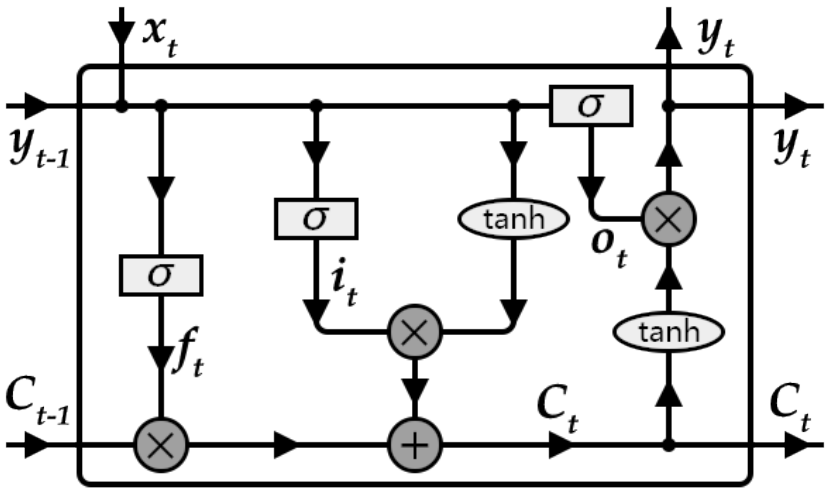


Figure 2.1: The internal structure of a single LSTM cell. Arrows indicate the direction of data flow.

Long short term memory networks and nonlinear autoregressive exogenous models are two types of RNNs we adopt in this paper. Both networks are designed to memorize historical data and identify trends. We explain them below.

Long Short Term Memory (LSTM) is a recurrent neural network which was first introduced in [56] and has been widely applied since then to learn long-term dependencies in data. As depicted in Figure 2.1, an LSTM cell is comprised of three gates: an input gate, an output gate, and a forget gate.

There is an identity activation function to compute the weight of the input. This weight decides whether the value is essential to be stored and sent to the output layer. Thus, it can memorize the value in an uncertain length of time.

We use the following notation to describe LSTM networks:

1. $d \in \mathbb{R}$: the dimension of the input vector
2. $h \in \mathbb{R}$: the dimension of the output vector
3. $t \in \mathbb{R}$: the time index
4. $x_t \in \mathbb{R}^d$: inputs at time t
5. $y_t \in \mathbb{R}^h$: outputs at time t

6. $f_t \in \mathbb{R}^h$: the activation vector for the forget gate at time t
7. $i_t \in \mathbb{R}^h$: the activation vector for the input gate at time t
8. $o_t \in \mathbb{R}^h$: the activation vector for the output gate at time t
9. $C_t \in \mathbb{R}^h$: the cell state vector at time t
10. σ: logistic sigmoid function
11. $tanh$: hyperbolic tangent function
12. W_f, W_i, W_o, W_C, b_f, b_f, b_i, b_o, b_C: weights and biases

The cell state is the memory in the module, enabling the LSTM network to memorize the values. The gates are designed to help the cell state vector to track useful information. There are three gates in each module. The result of the logistic sigmoid function is between 0 and 1. The result indicates the percentage of the value that should pass through.

The first (leftmost) gate in Figure 2.1 determines how much the input would affect the state, or equivalently, how much of the history would be forgotten given the current input. Therefore, the activation vector for the forget gate can be computed by:

$$f_t = \sigma(W_f \cdot [y_{t-1}, x_t] + b_f)$$

Here σ is the logistic sigmoid function which returns a value between 0 and 1.

Next, the input gate indicates which input (i_t) would update the state at time t, and the hyperbolic tangent function, denoted by $tanh$, creates the candidate values that could be added to the state. After these two gates, the state is updated as follows:

$$i_t = \sigma(W_i \cdot [y_{t-1}, x_t] + b_i)$$

After these two gates, we can update the cell state vector. We first need to forget some values by using the forget gate. We then add the new information into the vector by employing the input gate. The update function is defined as follows:

$$C_t = f_t \times C_{t-1} + i_t \times tanh(W_C \cdot [y_{t-1}, x_t] + b_C)$$

The last (rightmost) gate in the LSTM module computes the final output. The output is picked from the current state (passed through a *tanh*) using a Sigmoid gate to evaluate which value it should present in the output vector. The final output vector y_t can be computed as follows:

$$o_t = \sigma(W_o \cdot [y_{t-1}, x_t] + b_o)$$

$$y_t = o_t \times tanh(C_t)$$

The current output, y_t, is an input to the next LSTM module. We use the backpropagation-through-time algorithm to train this model.

The LSTM network used for occupancy modelling in this thesis has one hidden layer which contains 64 cells. The output layer contains one output node representing the occupancy state, and the input layer contains as many nodes as the number of features in our data set. The cost is computed using the softmax cross entropy between logits and labels. We partition data into multiple batches, where the batch size is equal to one day (from 12:00 am to 11:59 pm). For each batch, we minimize the cost using a first-order gradient-based optimization method [63].

Nonlinear Autoregressive Network with Exogenous Inputs (NARX) is a nonlinear autoregressive time-series model with one or more exogenous inputs influencing the output. It is a powerful model when it comes to discovering long-term dependencies. It can be used to model nonlinear dynamic systems and have been applied in time-series modelling [22]. We use the backpropagation-through-time algorithm to compute the gradient for calculating the weights.

The input to this model is the current and d past input values $(x_t, x_{t-1}, \cdots, x_{t-d})$, along with d past output values $(y_{t-1}, \cdots, y_{t-d})$, simulating the memory cell. Feeding previous outputs to the input layer allows for storing historical data and helps with learning the long-term dependencies. Figure 2.2 depicts the structure of an NARX network.

The NARX model uses the following notations.

1. t: the time label

2. T: the set of time labels

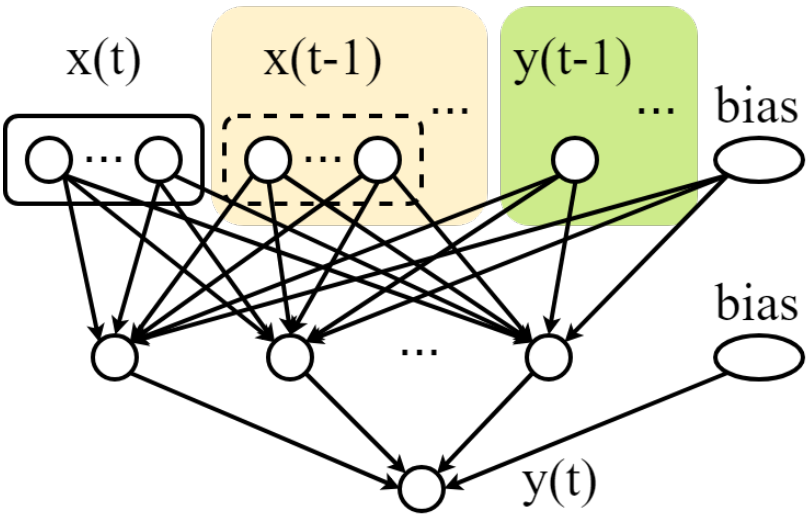


Figure 2.2: Schematic of an NARX network.

3. N: the set of neurons
4. O: the set of output neurons
5. I: the set of input neurons
6. $x(t)$: the inputs at time t
7. $y(t)$: the output at time t
8. C: the state variables

The state space representation of the NARX model can be written as [75]:

$$C_i(t+1) = \begin{cases} \Phi(u(t), C_i(t)), & i = 1 \\ C_i(t), & i = 2, \cdots, N \end{cases}$$

where $\Phi(\cdot)$ represents the nonlinear mapping of the neural network, $u(t)$ represents the past inputs at time t. The output at time t can be computed by $y(t) = C_1(t)$. The forgetting behavior is due to:

$$\lim_{m \to \infty} \frac{\partial C_i(t)}{\partial C_j(t - t')} = 0 \qquad \forall t, t' \in T, i \in O, j \in I$$

We use a NARX network with one hidden layer containing 40 nodes. We use the dynamic backpropagation algorithm to optimize ttimeserieshe cost. The model is similar to a simple feed-forward neural network [78].

2.2 Thermal modelling

Thermal modelling is essential for designing building envelops [11], conducting an energy audit, managing energy consumption of HVAC systems (through chiller sequencing and occupancy-based HVAC control [71] for example), and demand response programs [34]. The lumped parameter thermal model is one of the most common thermal models, which offers a better understanding of temperature dynamics of the building by dividing its interior and envelope into a number of temperature-uniform lumps. It models heat flux through the building by creating an analog circuit model [61]. Thus, this physics-based model is also known as the Resistance-Capacitance (RC) model.

Many software and tools rely on the RC models to simulate and analyze building energy use, such as Smart-E [18], CitySim [93], DIMOSIM [89], BuildingSystems [69], [81], and BuildSysPro [92]. They build arbitrarily complex RC models with several sub-circuits for different rooms and the building envelope.

In addition to RC thermal modelling, the temperature inside a building can be described using a time-series model. Compared to building RC models, this approach is completely data-driven rather than physics-based. It develops separate models for the temperature inside different rooms. The models developed for different rooms can be combined by referencing the output of other models as explanatory variables of the model.

Previous studies used various combinations of sub-circuits in the RC network, such as 4R1C [53], [71] and 3R2C [82], [110] for the envelope model, and 2R2C [82] and 1R1C [53], [110] for the room model. Reference [107] compared a low dimensional data-driven thermal model to a high dimensional physics-based one, concluding that the data-driven model can substitute the high-dimensional physics-based model with a negligible loss of accuracy.

Table 2.2 shows the taxonomy of related work on thermal modelling using various platform, model settings, parameter estimation methods, and model scales.

Table 2.2: Taxonomy of related work on thermal modelling

Type	Platform/Algo.	Envelope model	Room model	Parameter estimation	Model scale	Objective	Ref.
White-box	Smart-E	10 parameters	Utility models	Measurements	City	Demand response strategy	[18]
	RC model	3R2C	1R1C	Genetic algorithm [77]	Floor	Demand response strategy	[110]
	SHEMS	Various	1C	MILP	House	Demand response strategy	[34]
	CitySim	4R1C	1R1C	Analog value	Office room	Human-energy interaction	[53]
	DIMOSIM	6R3C	1R1C	Measurements	Building	Model comparesion	[89]
	BuildingSystems	3R2C	5R1C	Measurements	Thermal zone	Model comparison	[69]
	RC model	3R2C	N/A	Non-linear optimization	Room	RC parameter estimation	[54]
	RC model	3R1C	1C	Measurements	Freezer room	Simulate thermal behavior	[38]
	TRNSYS 17	Various	Various	Measurements	House	Suggest building envelope	[11]
	BEAM	3R2C	1C	Measurements	House	Suggest control state	[95]
Grey-box	RC Model	3R2C	2R2C	Genetic algorithm [77]	Thermal zone	Cooling load modelling	[82]
	RC model	4R1C	1C	Parameter adaptive building	Office room	Model uncertainty	[71]
	TRNSYS	10R6C	3R2C	Non-linear regression	House	Predict HVAC loads	[20]

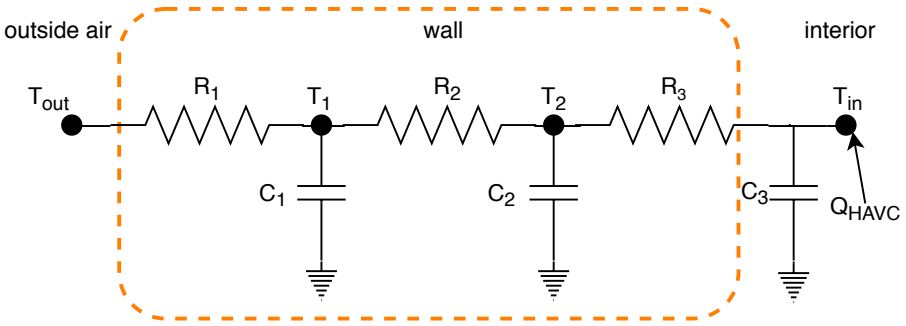


Figure 2.3: The lumped parameter thermal model representing the building envelope with a 3R2C network and its interior with a single capacitance, 1C. This simplified model incorporates the heat introduced or extracted by the HVAC system, but neglects internal heat gain and solar radiation.

2.2.1 RC-network models

The RC model describes heat flux, Q, in a building similar to how current flow is described in an electric circuit composed of several resistors and capacitors in series or shunt. Concretely, a thermal resistor in the RC model creates a difference in the temperatures of its two terminals. We have $Q = \Delta T/R$, where R is the thermal resistance, and ΔT represents the temperature difference. The thermal capacitor in the RC model acts as a thermal energy reservoir; it can store heat: $C\frac{\partial T}{\partial t} = Q$, where C is the thermal capacitance. The RC model divides the building space and its envelope into a certain number of lumps. The fundamental assumption here is that the temperature of each lump is uniform.

A variety of RC network models have been proposed in the literature. A high-order RC model is generally more accurate and more complex, but this complexity comes at a price: it is computationally expensive to use this model especially for fast timescale analysis of heat transfer and real-time control applications. A reduced-order thermal model can offer a reasonable tradeoff between accuracy and model complexity. For example, a 2nd-order RC model would give minimal loss of accuracy compared to a 20th-order benchmark model with a considerably lower computational demand [50]. Hence, in this work we focus on developing low-order RC network models.

We model the building envelope as a 3R2C network. This model is com-

prised of three thermal resistors, denoted by R_1, R_2 and R_3, and two thermal capacitors, denoted by C_1 and C_2, describing the three-layer structure of the wall [54], [95]. The building interior is modelled using a thermal capacitor (C_3) as it absorbs and retains heat. It is also connected to the HVAC system which injects or extracts heat from the building space based on the operation mode of the HVAC system. We denote the total heat flux from the HVAC system by Q_{HVAC}. Figure 2.3 shows the equivalent circuit of this RC model. A set of partial differential equations govern the dynamics of temperature evolution in the building space:

$$\begin{cases} \partial T_1 = \dfrac{1}{C_1 R_1}(T_{out} - T_1)\partial t + \dfrac{1}{C_1 R_2}(T_2 - T_1)\partial t \\ \partial T_2 = \dfrac{1}{C_2 R_2}(T_1 - T_2)\partial t + \dfrac{1}{C_2 R_3}(T_{in} - T_2)\partial t \\ \partial T_{in} = \dfrac{1}{C_3 R_3}(T_2 - T_{in})\partial t + \dfrac{Q_{HVAC}}{C_3}\partial t \end{cases} \tag{2.1}$$

where T_{in} represents the temperature inside the building, T_{out} represents the outside temperature, T_1 and T_2 represent the temperature of the two lumps used to model the wall, and $Q_{HVAC} = k_{heat}Q_{heat} - k_{cool}Q_{cool}$ where k_{heat} and k_{cool} are two binary control variables defined as:

$$\begin{aligned} k_{heat} &= \begin{cases} 1, & \text{if } T_{in} < T_{setheat} \ \&\ \text{HVAC}_{\text{mode}} \in \{auto, heat\} \\ 0, & \text{otherwise} \end{cases} \\ k_{cool} &= \begin{cases} 1, & \text{if } T_{in} > T_{setcool} \ \&\ \text{HVAC}_{\text{mode}} \in \{auto, cool\} \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{2.2}$$

Here $T_{setcool}$ and $T_{setheat}$ respectively denote the cooling and heating setpoints, Q_{cool} and Q_{heat} respectively denote the energy flux from cooling and heating systems, and $\text{HVAC}_{\text{mode}}$ denotes the operation mode of the HVAC system which can take a value from the following set: {*auto, heat, cool, off*}. When $\text{HVAC}_{\text{mode}}$ is *auto*, the HVAC system can be either in the heating or cooling mode depending on how the indoor temperature compares with the two setpoints. We assume the temperature sensor is installed in an ideal location in the room and can measure the room temperature with negligible error. We further assume that the HVAC system delivers (or extracts) the same amount of heat during the period it is heating (or cooling) the space.

Although there exists a large body of work that used RC models to study the building thermal transfer, most of them focus on modelling the heat flux using measurements of inside and outside temperature assuming that the resistance and capacitance parameters are known [20], [31], [38], [98]. In other words, they assume the knowledge about the building thermal mass and its insulation, which might be easy to obtain for commercial buildings, but is not necessarily available for residential buildings in operation today.

Since the resistances and capacitances depend on the construction material and building structure, they are treated as time-invariant or constant parameters. Our goal is to estimate these parameters using pre-trained neural network models so that the resulting RC model can be used to predict T_{in} given T_{out}, k_{heat}, and k_{cool}. This process is described in Section 5.1. Note that we ignore solar radiation and internal heat gain since this data is not provided in the data set we use.

2.2.2 Time-series models

The temperature inside a building is a time-series, and therefore, can be modelled using a standard time-series modelling technique. Time-series models relate the next observation to historical observations.

Autoregressive Integrated Moving Average (ARIMA) model is a time-series model which has been widely used in demand forecasting and various planning problems [26]. It can capture lagged correlations between forecast errors and time-series data. The ARIMA model is expressed as:

$$\left(1 - \sum_{i=1}^{p} \alpha_i B^i\right) (1 - B)^d X_t = \left(1 + \sum_{j=1}^{q} \beta_j B^j\right) \epsilon_t + c \tag{2.3}$$

where X is the time-series, $BX_t = X_{t-1}$, and p, d, q are the parameters of the ARIMA model. This model has three parts, namely the autoregressive part (AR), the integrated part (I), and the moving average part (MA). An ARIMA model is uniquely characterized by three parameters, p, d, q, corresponding to the three parts of the model, and coefficients of the AR and MA parts (α and β).

A time-series must be at least wide-sense stationary so that we can fit a suitable Autoregressive Moving Average model to it. When the time-series is not wide-sense stationary, taking differences of consecutive observations makes it stationary. Hence, differencing is the first step of building an ARIMA model for a non-seasonal time-series and corresponds to the integrated part of the model. The differenced time-series can be written as:

$$X'_t = X_t - X_{t-1}$$

where X_t and X'_t are respectively the value of the original time-series and the value of the differenced time-series at time t. If the data does not appear to be stationary after the first differencing we can perform differencing repeatedly until the resulting time-series is stationary. The number of differencing operations performed on the time-series is reflected in the parameter d. In practice, the second order differencing usually suffices.

Time-series may exhibit temporal correlation. In this case, past observations can be used to determine the next value. The autoregressive (AR) part of the model captures this intuition. Specifically, it defines the current value X_t as the weighted sum of the past values $X_{t-1}, \cdots, X_{t-p}$:

$$X_t = c + \sum_{i=1}^{p} \alpha_i X_{t-i} + \epsilon_t$$

Here c is a constant value, ϵ is the white noise term, α represents the weights, and p indicates the number of historical values used.

Lastly, the moving average (MA) part portrays the relationship between the next time-series value and historical white noise terms. The model estimates the current value X_t as the weighted sum of the past white noise terms:

$$X_t = \mu + \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \epsilon_t$$

Here μ is the mean of the series (often assumed to be 0), β represents the weights, and q indicates the number of historical white noise terms used.

Putting the three parts together, we obtain the model expressed in Equation 2.3. This model is particularly useful to model a stochastic process, even

when it is *non-stationary*. It is also suitable for modelling *non-seasonal* time-series, where the trend is not white noise and has a certain pattern.

In many cases, the time-series value depends on the values of some external or exogenous variables. The ARIMAX is a variation of the ARIMA model that takes these explanatory variables into account:

$$\left(1 - \sum_{i=1}^{p} \alpha_i B^i\right) (1 - B)^d \left(X_t - \sum_{k=1}^{r} \gamma_k E_t^{[k]}\right) = \left(1 + \sum_{j=1}^{q} \beta_j B^j\right) \epsilon_t + c$$

where $E_t^{[k]}$ is the value of k-th exogenous variable at time t, γ represents the weights of exogenous variables, and r is the total number of exogenous variables in the model. Moreover, some time-series data have seasonality, meaning that they exhibit a certain cycle or periodicity. To model seasonality, it is crucial to separate it from the original time-series. SARIMAX is an ARIMAX model that performs seasonality decomposition. It can be written as

$$\left(1 - \sum_{i=1}^{p} \alpha_i B^i\right) \left(1 - \sum_{i'=1}^{P} \alpha'_i B^{si'}\right) (1 - B)^d (1 - B^s)^D \left(X_t - \sum_{k=1}^{r} \gamma_k E_t^{[k]}\right) =$$

$$\left(1 + \sum_{j=1}^{q} \beta_j B^j\right) \left(1 + \sum_{j'=1}^{Q} \beta'_J B^{sj'}\right) \epsilon_t + c$$

where P, D, and Q are respectively the orders for the seasonal autoregression, seasonal integration, and seasonal moving average components, and s represents the seasonal length. A detailed explanation of how to estimate the parameters of the SARIMAX model is described in Section 5.1.2.

2.3 Transfer learning

Humans have the ability to transfer knowledge from tasks in which they had some experience to a new task, which may not be identical to these tasks. This past experience often helps them finish the new task more efficiently, even with less experience in that task. Drawing on this idea, "transfer learning" is introduced in the machine learning literature. As put by Andrew Ng, a renowned machine learning scientist, "after supervised learning, transfer learning will be the next driver of ML commercial success".

Training accurate data-driven models requires enormous amounts of data; this limits their application in practice. But transfer learning enables model training even when little or no training data is available. This is accomplished by adapting and reusing a *pre-trained* model.

The initial idea of transfer learning came out as early as the 1990s. The most attractive part of transfer learning is the ability to solve a complex problem starting with an almost optimal model. This means that the amount of training data can be reduced significantly, just as the time and effort to train a suitable model. Moreover, even if people are willing to collect and train such complex models, in the real world scenario, data is usually difficult to collect, and people cannot directly reuse well-trained models in a new environment. For example, it took several years to build ImageNet [32].

The formal framework of transfer learning is defined by [85]. The framework defines a *domain*, denoted by $\mathbf{D}$, as the combination of feature space $\mathbf{X}$ and marginal probability $P(\mathbf{X})$, i.e., $\mathbf{D} = \{\mathbf{X}, P(\mathbf{X})\}$. Also, the framework states a *task*, denoted by $\mathbf{T}$, as the combination of label space $\mathbf{Y}$ and objective function $P(\mathbf{Y}|\mathbf{X})$, i.e., $\mathbf{T} = \{\mathbf{Y}, P(\mathbf{Y}|\mathbf{X})\}$. Therefore, given the source domain $\mathbf{D}_S$, task $\mathbf{T}_S$, target domain $\mathbf{D}_T$, and task $\mathbf{T}_T$, transfer learning suggests a good prior of $P(\mathbf{Y}_T|\mathbf{X}_T)$ in $\mathbf{D}_T$ with the knowledge from $\mathbf{D}_S$ and $\mathbf{T}_S$, where $\mathbf{D}_S \neq \mathbf{D}_T$ or $\mathbf{T}_S \neq \mathbf{T}_T$.

We classify transfer learning techniques into four classes: domain adaptation, domain confusion, one-shot learning, and zero-shot learning. Note that these classes are not mutually exclusive because a transfer learning technique can belong to more than one of these classes.

2.3.1 Domain adaptation

Domain adaptation usually refers to the case of where marginal probabilities are different between domains, i.e., $P(\mathbf{X}_S) \neq P(\mathbf{X}_T)$. It seeks to learn from one or multiple *source domains* a model that performs well on a related *target domain*. It is assumed that the source and target domains are associated with the same label space. Domain adaptation has been used extensively in computer vision and natural language processing [104]. The early applications

of domain adaptation can be traced back to 1990s [21]. In recent years, domain adaptation has been applied to the image translation problem. Reference [109] develops two conditional Generative Adversarial Networks (GANs), one to translate an image from the source domain to the target domain ($Y \leftarrow f(X)$), and another one to translate it from the target domain to the source domain ($X \leftarrow g(Y)$). The two networks are trained to minimize the difference between X and $g(f(X))$, enabling translations such as a zebra to a horse or a photo to a Monet painting. Reference [62] transfers both texture and geometrical properties of an image, enabling them to successfully transform a chair to a car or a vehicle to a human face. Reference [44] employs a domain adaptation technique to build a Convolutional Neural Network (CNN) model for image recognition on a large set of car images drawn from e-commerce websites and Google Street View. In a different line of work, Reference [30] uses domain adaptation to find the common embedding between two languages to perform an accurate translation.

Despite the extensive literature on domain adaptation, little work has been done to investigate whether it can be applied to data collected by IoT devices which are possibly deployed in different environments. To our knowledge, Reference [9] is the only paper that utilizes domain adaptation to determine the number of occupants in a room by using carbon-dioxide measurements from this room and a large cinema. It proposes a human occupancy counter which employs an accurate occupancy model trained with minimum labelled data. This model is developed in a small room and then adapted to a larger room (a cinema with a seating capacity of 279 people). The authors develop a seasonal decomposition model which captures the nonlinear relationship between carbon dioxide and occupant count. This model has four components which are trained in the source domain and then adapted to the target domain. They show that it is possible to achieve higher accuracy with the adapted model on the target domain. The accuracy they report is around 90% for binary occupancy detection and 60% for estimating the number of occupants. The main issue with this work is that they consider an occupancy estimation as accurate if the absolute differences between predicted and real occupant counts are less

than five people.

Our work on occupancy modelling is similar to [9] in that we also leverage semi-supervised domain adaptation to estimate the number of occupants with minimum labelled data from the target domain. However, their approach has several shortcomings. First, they only consider one feature, which is the carbon dioxide concentration, and build a model that only works if this feature is available; hence, it cannot be extended to other occupancy-indicating features. The carbon dioxide sensor is not always available in the HVAC system, and it detects occupancy events after a certain delay since carbon dioxide builds up slowly. In our test, the carbon dioxide sensor takes about 15 minutes to sense any change in the carbon dioxide concentration level after an occupancy event. Second, they only study the case where the source and target domains have the same feature space. Third, the accuracy of their model is low when it comes to determining the number of occupants, especially given that they consider an occupancy estimation as accurate if the absolute differences between predicted and real occupant counts are less than five people. We address these shortcomings in this work, build recurrent neural network models that are general enough to be used with an arbitrary set of features, and evaluate our algorithm in two commercial buildings with different features located in two countries.

2.3.2 Domain confusion

The basic idea of domain confusion [42], [43] is to add another objective (or change the task in the source domain) that maps the source domain to a latent domain which is more similar to the target domain than the source domain. This change of the objective function in the source domain confuses the domain itself. Figure 2.4 shows the idea of reversing the gradient of the flow from the loss to the remaining network. The reversed gradient asks the model to maximize the error instead of minimizing the error which is the case in most networks. This allows the model to minimize the source objective and prevents it from differentiating the two domains.

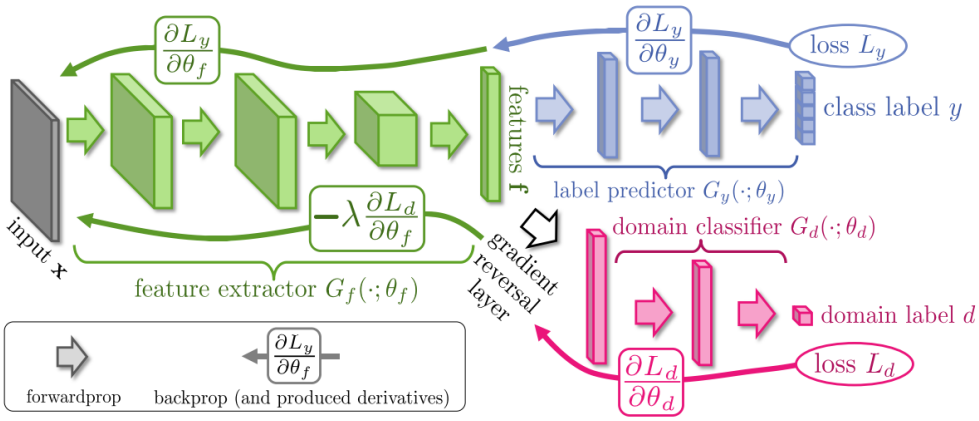


Figure 2.4: Confusing domains with a gradient reversal layer [42]

2.3.3 One-shot & zero-shot learning

There are two main approaches in transfer learning [48]: one-shot and zero-shot learning. Consider a model that is trained in a given source domain. In one-shot learning, some amount of labelled data from the target domain is utilized to retrain this model, whereas in zero-shot learning this pre-trained model is used without adaptation to make predictions in the target domain. We compare the accuracy of models obtained through one-shot learning and zero-shot learning in Section 3.3 and Section 5.2.

2.4 Open platforms and toolkits

An inevitable part of scientific research is developing mathematical models and algorithms. Each model serves a specific purpose and is often tailored for a target application. Thus, most models cannot be generalized and used in a different context. This has made model validation an essential task. Researchers need a set of tools to handle different types of input data and are required to write many lines of code to compare their new model with existing models.

To address this problem, a publicly available toolkit would be a natural choice. A lot of work in literature offers publicly available toolkits for different purposes. NILMTK [16] is an open-source toolkit that provides an easy to use platform for evaluating different energy disaggregation algorithms across differ-

ent data sets. It contains tools for analyzing and processing publicly available data sets, implementation of well-known energy disaggregation algorithms for benchmarking, and a diverse set of evaluation metrics. scikit-learn [87] is a well-known toolkit that incorporates a vast number of machine learning algorithms with various data sets and analysis APIs. GraphLab [70] is another robust machine learning framework that provides a distributed as well as unified multicore API for parallel machine learning algorithms. For physiologic signal processing and analysis, PhysioToolkit [46] is an API library with an extensive collection of algorithms and tools. For wireless network community, CRAWDAD [65] compiles several data sets and algorithms.

We argue that a similar toolkit is needed for evaluating occupancy detection models on publicly available data sets using standard metrics. The design and implementation of this toolkit are discussed in Chapter 4.

Chapter 3

Fine-Grained Occupancy Estimation

Fine-grained occupancy information is essential to improve the human experiences and operational efficiency of buildings, yet it is quite challenging to obtain this information due to the lack of special-purpose sensors for occupancy monitoring, and insufficient training data for developing accurate data-driven models. This chapter addresses this challenge by (a) utilizing recurrent neural network models to uncover latent occupancy patterns in individual rooms from trend data available through the building management system, and (b) applying a domain adaptation technique to transfer existing occupancy models trained in a controlled environment (i.e., the source domain) to another environment (i.e., the target domain) where labelled data is sparse or non-existent. We adjust the model parameters based on the apparent differences between the two environments and apply the adapted model to estimate the number of occupants in the target domain. Our results from two commercial test buildings in two continents indicate that the adapted model yields only slightly lower accuracy than a model that is built initially on the target domain gave a large amount of labelled data. Furthermore, we study how many labelled data is required from the target domain for the semi-supervised domain adaptation technique to achieve promising results.

Our approach is to build data-driven occupancy models using the trend data, e.g., measurements of carbon dioxide and damper position sensors, which are readily available through the BMS in most commercial buildings. Since

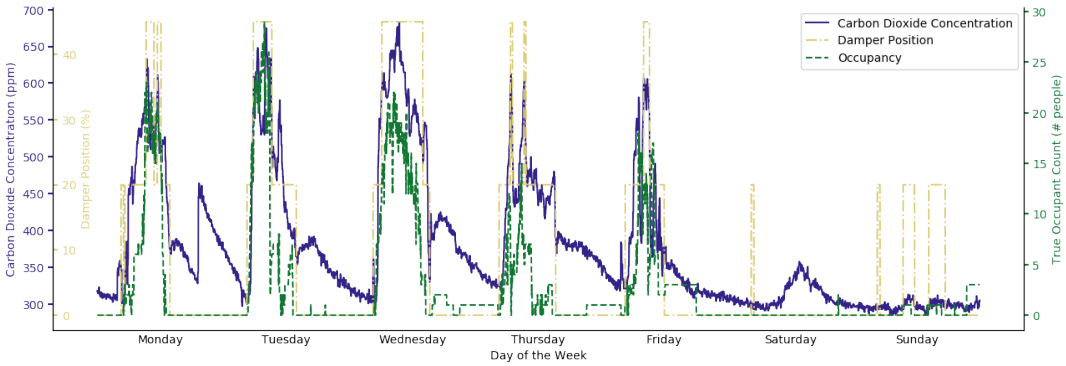


Figure 3.1: Ground truth occupancy data and measurements of carbon dioxide and damper position over one week in a room in Building A.

training these models requires an abundance of labelled occupancy data, we investigate the use of a semi-supervised domain adaptation technique to transfer occupancy models that are built in a controlled environment where sufficient labelled data is available to an environment where little or no labelled data is available. We study this problem when the source and target domains are in the same building.

The rest of this chapter is organized as follows. Section 3.1 shows the feasibility of estimating the room-level occupant count from the available trend data. Section 3.2 describes our data sets and defines the proposed methodology for training and adapting the occupancy models. Section 3.3 explains the evaluation results, and Section 3.4 presents discussion points and suggests directions for future work.

3.1 HVAC system and trend data

An HVAC system typically consists of one or more Air Handling Units (AHUs), which supply cold air through ducts to Variable Air Volume (VAV) systems, each controlling a thermal zone. If a zone requires cooling to balance the heat gained from occupants, appliances, and external sources, the VAV unit opens its dampers to the required extent to allow cooler air to flow into the zone. Conversely, if a zone requires heating to maintain its operating point, the VAV unit opens the radiator or reheat valve.

The heating or cooling action of a VAV unit is determined by a control

system which keeps the zone temperature around its setpoint while maintaining the required minimum airflow. The VAV control system monitors the zone temperature and actuates the dampers and valves. The BMS typically logs the instantaneous values of the sensors and states of the actuators.

Figure 3.1 shows the ground truth occupancy (i.e., the number of people in the room), the carbon dioxide concentration level, and the damper position over one week in a room in our test building (as described in Section 3.2.1). It can be readily seen that the number of occupants is significant when the carbon dioxide concentration peaks and the damper becomes half-open. Furthermore, it can be seen that the carbon dioxide concentration is low, and the damper is closed most of the time during the weekend when the room is generally unoccupied. This implies that these sensors are occupancy-indicating, and it is possible to uncover the underlying occupancy pattern of a room using only trend data available through the BMS and sophisticated machine learning algorithms.

3.2 Methodology

Our goal is to transfer an existing model built for a room with sufficient training data to another room in the same building, for which training data is sparse or non-existent. Our hypothesis is that the adapted model is more accurate than a model that is originally built on the target domain using limited training data that is available. In the following, we describe our data set and then present our domain-adaptive occupancy models.

3.2.1 Data set

Our data set is comprised of two commercial buildings located in two countries, which are referred to as Building A and Building B, respectively. Both buildings are equipped with a BMS system capable of logging, trending, and reporting. Building A is a $8,500m^2$ campus building at the University of Southern Denmark [6] with an average of 1,000 occupants on normal weekdays. The building contains graduate student and faculty offices, lecture rooms, and

Table 3.1: Description of Building A

Room type	Study area		Lecture room	
Room number	**1**	**2**	**3**	**4**
Seating capacity	36	36	85	85
Var. occupancy	21.7	20.5	67.7	231.9
PAR occupancy	13.7	10.4	10.6	9.0
Min. CO_2 level (ppm)	256	268	370.88	304
Max. CO_2 level (ppm)	907.52	688	844.8	1384
Area (m^2)	125	125	139	139
Max. air flow (m^3/h)	3000	3000	4800	4800

Table 3.2: Description of Building B

Room type	Meeting room		
Room number	**A**	**B**	**C**
Seating capacity	10	8	10
Var. occupancy count	3.35	1.25	2.27
PAR occupancy count	107.1	89.3	95.1
Min. temperature (°C)	20.10	20.81	20.74
Max. temperature (°C)	27.76	25.97	28.39
Relative area	1.2	1	1.2
Number of windows	2	0	2
Max. air flow (L/s)	130	85	130
Min. air flow (L/s)	65	6	65

study areas. We consider four rooms in this building to develop and validate the data-driven occupancy models. Each rooms is equipped with high-precision people counting cameras mounted over the two entrances to record the number of occupants, which is treated as ground truth occupancy data. Two of those rooms, Room 1 and Room 2, are $125m^2$ study areas with the seating capacity of 36 people. The other two rooms, Room 3 and Room 4, are $139m^2$ lecture rooms with the seating capacity of 85 people. The HVAC system in this building supplies a maximum of $3000m^3/h$ and $4800m^3/h$ fresh air into the study areas and lecture rooms, respectively. Table 3.1 summarizes the information about these rooms, including the variance and peak-to-average ratio

(PAR) of the number of occupants during the period that data was collected.

Each room constitutes a thermal zone and is controlled by a separate VAV system. The VAV system uses two sensors, i.e., carbon dioxide and damper position sensors, in each room to control the indoor climate. The carbon dioxide sensor samples the carbon dioxide concentration level in parts per million (ppm), and the damper position sensor measures the damper openness in percentage of fully opened. Both quantities are sampled at one minute intervals and measurements are archived by the BMS system. The measurements are obtained between March 21st, 2017 and April 6th, 2017 through the API of the BMS system. The ground truth data was also available for the whole period.

Building B is a four-story office building owned and operated by PCL Constructors in Edmonton, AB, Canada. It contains $44,000 ft^2$ of office space, workstations, and meeting rooms. This building does not have a vision-based system for collecting ground truth occupancy data. Thus, we only consider three meeting rooms for which we could extract the number of attendees, time, and duration of meetings from their calendars. Admittedly, the obtained ground truth data is not reliable, because people may enter the room before the meeting starts and may leave before it ends. Moreover, the real number of attendees may be different from the number of people accepted the calendar invitation. In any case, this data shows the overall occupancy trend in each meeting room. Table 3.2 summarizes the information about these rooms. Each meeting room constitutes a thermal zone and is controlled by a separate VAV system. Each VAV system has several sensors in each room to control the indoor climate. We obtained temperature, airflow, pressure, and damper position data sampled by these sensors at 10-minute intervals. We obtained measurements between December 18th, 2016 and December 18th, 2017 through the BMS system.

Room A and Room C are corner meeting rooms on two consecutive floors of the building with the exact same size and layout; they have floor-to-ceiling glass on the north and east sides, and 4 diffusers. Room B is an interior room with no window and is on the same floor as Room A, but has a different layout. It has 2 diffusers. All meeting rooms have a rectangular conference table with

chairs arranged around it on all four sides. The VAV systems of these three rooms are connected to the same AHU.

In addition to the trend data, we extend our feature space by acquiring cloud cover and outdoor temperature for the cities where Building A and Building B are located using the Dark Sky API [101]. In each case, the weather and climate data are available every one minute during the intervals that sensor data was collected. We consider these features when developing a model for perimeter rooms that have at least one window. This is because the outdoor temperature and cloud cover (as a proxy for solar irradiance) can affect the heat load in the room, thereby causing the HVAC system to respond, for example by opening or closing dampers. It is important to make sure that these effects are not confused with the heat gain due to occupants.

3.2.2 Preprocessing

Our data set contains noisy and missing values and must be cleaned before it can be used to build occupancy models. We specifically identify and remove redundant time value pairs, resample all features at the same frequency, and impute missing values using a simple linear interpolation of neighboring, non-missing values. Once the data are cleaned, we combine all features for each time slot and store them in an array data structure.

3.2.3 Data-driven models for occupancy estimation

We develop four data-driven models to predict the number of occupants in the source domain where sufficient ground truth data (labelled occupancy data) is available. To this end, we train LSTM and NARX networks introduced in Section 2.1.2. These nonlinear models that have memory, are suitable for uncovering latent occupancy patterns. We also adopt Support Vector Regression (SVR) and Logistic Regression (LR) to estimate the number of occupants. These models are used to evaluate our proposed recurrent neural network models.

To understand how these models would generalize to an independent data set, we use 5-fold cross-validation. In particular, we split data from the source

domain into five equal sized segments, and use four of them (80% of data) to train the model, and the rest (20% of data) to test it. This process is repeated five times with different segments selected for testing. We compute the Root-Mean-Square Error (RMSE) to score the models and set model parameters to the ones that had the smallest RMSE. In addition, we calculate the normalized RMSE (nRMSE) which is the ratio of the RMSE to the range of occupant counts in the corresponding room. These metrics are defined as follows:

$$RMSE = \sqrt{\sum_{t \in T} (\hat{y}_t - y_t)^2},$$

$$nRMSE = \frac{\sqrt{\sum_{t \in T} (\hat{y}_t - y_t)^2}}{\max y - \min y},$$

where y denotes the true number of occupants in a given room and $\hat{y}$ denotes the predicted number of occupant in that room. We note that we do not use Mean Absolute Percentage Error (MAPE) as an evaluation metric in this case since it not defined when the room is not occupied (it results in division by zero). Instead, we use nRMSE which also gives a relative sense of how accurate the occupancy estimations are.

3.2.4 Semi-supervised domain adaptation technique

We now present a domain adaptation technique to build a model for estimating the number of occupants in a given room. We assume that the true number of occupants is known in the source domain for the entire duration that trend data is available. Hence, we use all trend data and corresponding occupancy labels for model training. Domain adaptation transfers this model to the target domain where ground truth data is limited or non-existent. The adapted model is then used for classification or regression in the target domain.

We leverage both semi-supervised and unsupervised domain adaptation techniques. Both techniques update the model trained in the source domain based on the differences between the source and target domains. The difference is that the unsupervised domain adaptation technique applies the model trained in the source domain to estimate the number of occupants in the target domain without updating the model using labelled data from the target do-

main, whereas the semi-supervised domain adaptation technique utilizes the available labelled data from the target domain to retrain the model. This retraining phase is key especially when the source and target domains have different feature spaces.

Figure 3.2 shows the overall framework for developing occupancy models. Since sufficient ground truth data is available in the source domain, it is possible to develop a well-suited model using supervised learning. Turning our attention to the target domain, one can develop a model using supervised learning but since labelled data is sparse, if not non-existent, the accuracy of such a model will not be high. Alternatively, it is possible to apply the supervised learning model built on the source domain to estimate the number of occupants in the target domain. We refer to this approach as unsupervised domain adaptation. Lastly, it is also possible to use a semi-supervised domain adaptation technique to adapt the model trained in the source domain to the target domain. This is a two-step process which starts with re-weighting the model and then retraining it using a small amount of labelled data that is otherwise insufficient for training an accurate model. We explain these two steps below.

Re-weighting:

The first step in our domain adaptation technique is called re-weighting. The basic idea is to compute transform matrices A_w and A_b, extract the weights w and biases b from the model, and finally use $A_w w$ and $A_b b$ as the new weights and biases, respectively. When the target domain is identical to the source domain, the transform matrices, A_w and A_b, are identity matrices. Otherwise, we need to construct A_w and A_b based on the apparent differences between the two domains, including size, seating capacity, and maximum air flow (or ventilation power).

The damper position determines how much fresh air can enter the room. Hence, the higher the maximum air flow is, the larger the effect of the damper position is on the amount of supplied air. Thus, the weight corresponding to

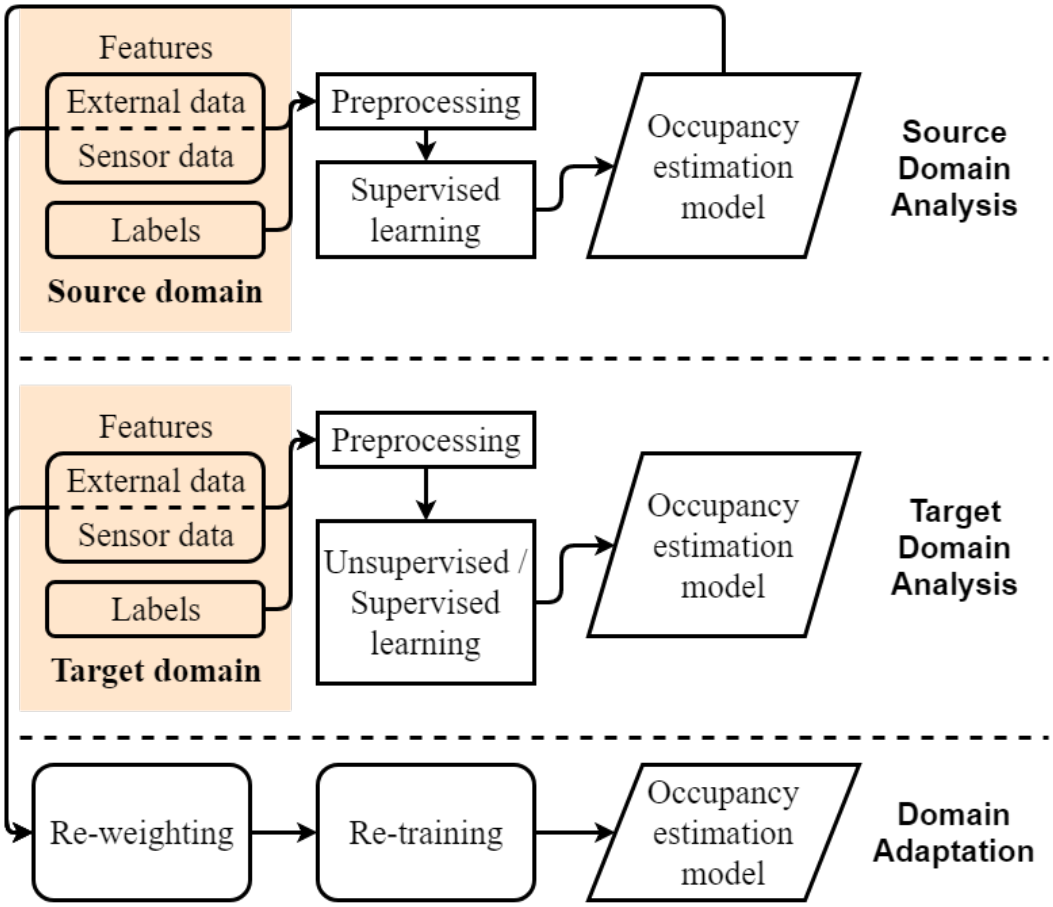


Figure 3.2: Data collection and analysis framework

the damper position in the input node must be updated based on the ratio of the maximum air flow in the target domain to the maximum air flow in the source domain:

$$w_{damper\_in} \leftarrow w_{damper\_in} \cdot \frac{\text{Max. airflow}_{\text{target}}}{\text{Max. airflow}_{\text{source}}}$$

We only change the weight of the node corresponding to the damper position. The weights of all other nodes remain the same.

As the size of the room and the maximum air flow change, the weights of all gates that are related to the carbon dioxide concentration level need to be adjusted. The unit of the air flow is m^3/h which is the volume of fresh air that enters the room per hour. Suppose all rooms within a building have the same height. We have;

$$\text{Room's Volume} = \text{Height} \times \text{Floor Area},$$

$$\text{Time} = \frac{\text{Room Volume}}{\text{Air flow}},$$

where *Time* is approximately the amount of time that it takes to fill the room with fresh air. We use this value to estimate the carbon dioxide diffusion rate. When it is smaller, it implies that carbon dioxide diffuses faster, hence the carbon dioxide sensor would measure it faster. In this case, we must increase the weight of the carbon dioxide feature. But unlike the damper position, which varies between 0% and 100%, the carbon dioxide level does not have a fixed upper bound. This means that we should not change the input weight of the carbon dioxide sensor. Instead, we increase the weight in each gate that corresponds to the carbon dioxide feature. In particular, w_{co_2} must be updated as follows:

$$\begin{aligned} w_{co_2} &\leftarrow w_{co_2} \cdot \frac{\text{Time}_{\text{source}}}{\text{Time}_{\text{target}}} \\ &\leftarrow w_{co_2} \cdot \frac{\text{Height} \cdot \text{Area}_{\text{source}}/\text{Max. airflow}_{\text{source}}}{\text{Height} \cdot \text{Area}_{\text{target}}/\text{Max. airflow}_{\text{target}}} \\ &\leftarrow w_{co_2} \cdot \frac{\text{Area}_{\text{source}} \cdot \text{Max. airflow}_{\text{target}}}{\text{Area}_{\text{target}} \cdot \text{Max. airflow}_{\text{source}}} \end{aligned}$$

Moreover, we need to adjust the bias terms, b, when we adjust the weights. We do this based on the differences in the seating capacity of source and target domains because the HVAC system is designed to condition each room based on its maximum occupancy. We have:

$$b \leftarrow b \cdot \frac{Capacity_{target}}{Capacity_{source}}.$$

Figure 3.3 shows the weights of an RNN cell that we update in the re-weighting progress; these weights are represented with dotted lines.

Retraining:

After re-weighting the model, we train it again using the available labelled data from the target domain to update the weights. We choose this training data from a contiguous time period so as to preserve the temporal dependency in the occupancy data. We use the same algorithm used to train the neural

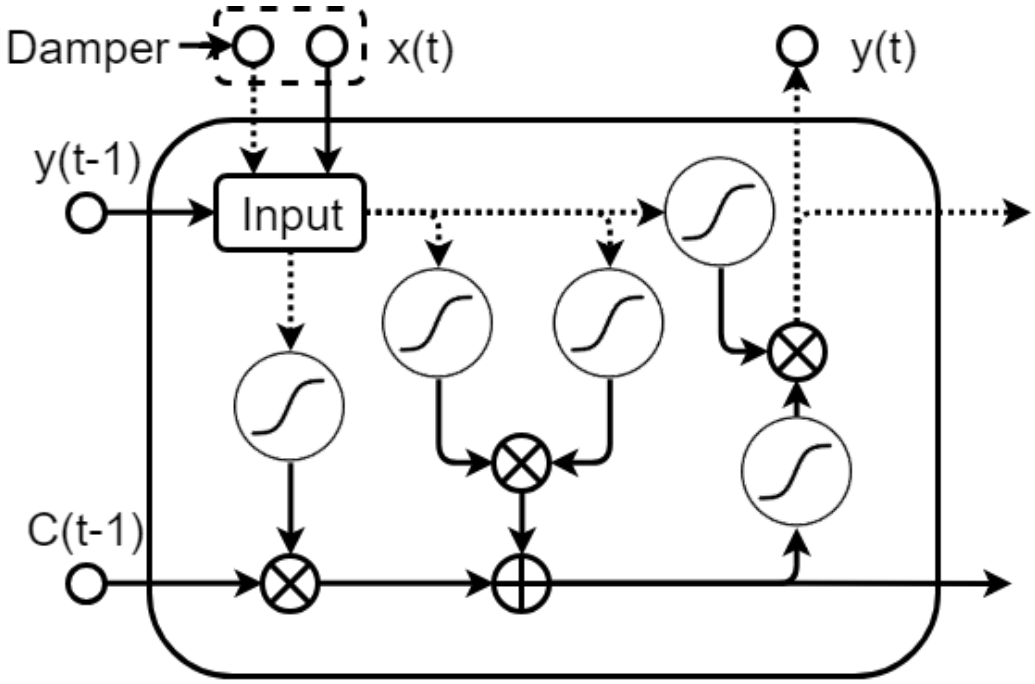


Figure 3.3: An illustration of the domain adaptation process. Dotted lines represent the weights that are updated.

network model to retrain it, the only difference being that the weights are initialized to the weights that are calculated in the re-weighting step.

3.2.5 Post-processing

To prevent error propagation over time, we correct obvious erroneous predictions in the post-processing step. This includes a negative or an unreasonably large number of occupants, and drastic changes in the number of occupants. We say that the predicted number of occupants is 'unreasonably large' when it is greater than the seating capacity of the room plus Δ. We use a non-negative value for Δ to account for the cases that the number of occupants is temporarily above the seating capacity of the room (e.g., between two successive classes when some students are entering the room while others are leaving it). We refer to changes in the occupant count as 'drastic' when the occupancy count increases or decreases by more than K occupants in an interval of length T. We set the values of Δ, K, and T of a given room based on its type, size, and function. In our experiments, we set Δ to 5. For study areas, we set K to 5

occupants and T to 4 minutes. For lecture rooms, we set K to 10 occupants and T to 1 minute.

To correct these errors, we replace negative occupant counts by zero, and unreasonably large occupant counts by the seating capacity of the room plus Δ. Moreover, drastic changes in the occupant count are replaced by K.

3.3 Results

In this section we corroborate the efficacy of the proposed semi-supervised domain-adaptive models by comparing them with two supervised learning models developed for occupancy estimation. We run each model 10 times for a given parameter setting, compute the mean and standard derivation of its prediction accuracy, and report the 97% confidence interval. The results presented in this section belong to Building A unless otherwise stated.

3.3.1 Evaluating the model trained using semi-supervised domain adaptation

Figure 3.4 shows the accuracy of estimating the number of occupants using different techniques and models. This figure is divided into three parts to group supervised learning models, unsupervised domain adaptation models, and semi-supervised domain adaptation models. The labels below the x-axis show whether re-weighting is carried out for semi-supervised and unsupervised domain adaptation techniques. We choose the best parameters for each model. We assume that labelled data is available only for 1 hour from the target domain. The target domain is Room 1 and Room 2 and the source domain is Room 3 and Room 4 in Building A. The supervised learning models are trained using 1 hour of labelled data from the target domain only, ignoring labelled data from the source domain.

It can be readily seen that the supervised learning models have the lowest accuracy. For example, the supervised LSTM model has an RMSE of 6.1094 (nRMSE of 17%), which is 43.76% higher than that of the LSTM model trained with the semi-supervised domain adaptation method. Moreover, the

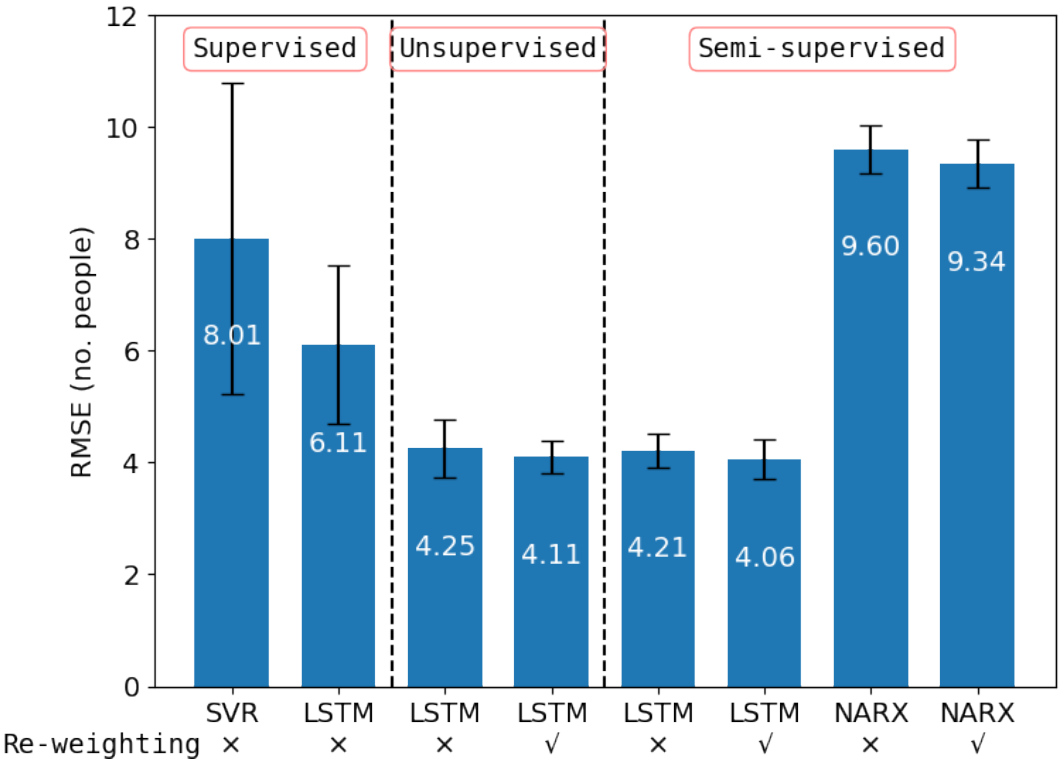


Figure 3.4: Comparing the accuracy of different supervised, semi-supervised, and unsupervised learning algorithms. Error bars represent the 97% confidence interval.

LSTM model has a significantly higher accuracy than the NARX model. The LSTM model trained with the unsupervised domain adaptation method on the target domain without re-weighting has the worst performance among all LSTM models built using domain adaptation. The LSTM model trained by the semi-supervised domain adaptation technique with re-weighting has the highest accuracy. In particular, it achieves an RMSE of 4.0599 (nRMSE of 11%) which is 1.23% lower than that of unsupervised domain adaptation with re-weighting and 3.69% better than that of semi-supervised domain adaptation without re-weighting.

In conclusion, Figure 3.4 shows that domain adaptation can help to increase the accuracy when the ground truth labels are insufficient in the target domains. In the meantime, semi-supervised domain adaptation is better than unsupervised domain adaptation, and our re-weighting method can also reduce the errors. Moreover, the re-weighting can decrease more errors than

Table 3.3: Comparing the performance of supervised learning models trained using labelled data only from the target domain in terms of their average and standard deviation of RMSE, and nRMSE.

		Training data		
		1 hour	**3 hours**	**1 day**
SVR	**RMSE**	8.01	***5.88***	***4.39***
	nRMSE	0.22	***0.16***	***0.12***
	STD.DEV	2.79	1.33	0.68
LSTM	**RMSE**	***6.11***	5.99	4.67
	nRMSE	***0.17***	0.17	0.13
	STD.DEV	1.41	1.23	0.41
Logistic Regression	**RMSE**	\	\	4.89
	nRMSE	\	\	0.14
	STD.DEV	\	\	0.36

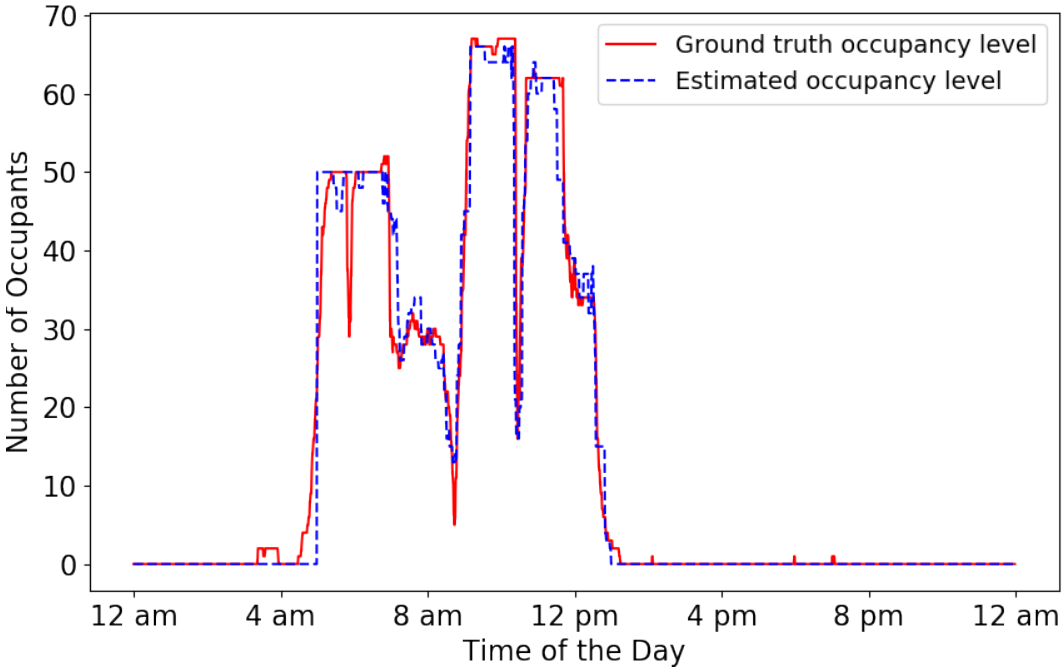


Figure 3.5: Estimated and true occupancy levels of a lecture room in Building A over one day.

changing the unsupervised learning to semi-supervised learning. All of those results show that our approach is the best approach when the ground truth labels in the target domains are limited.

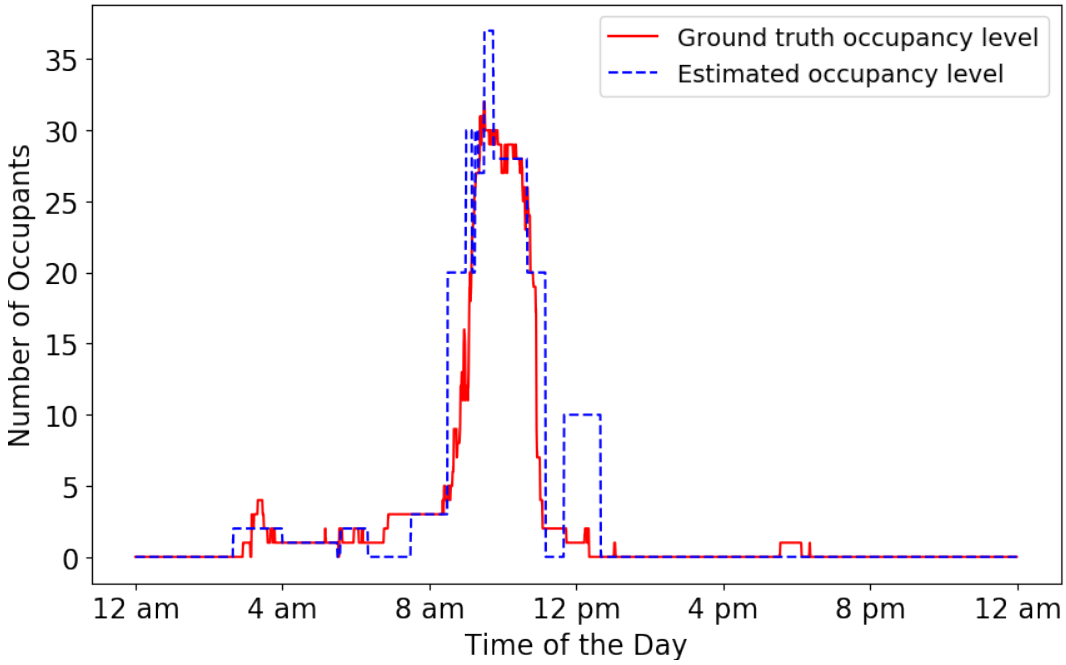
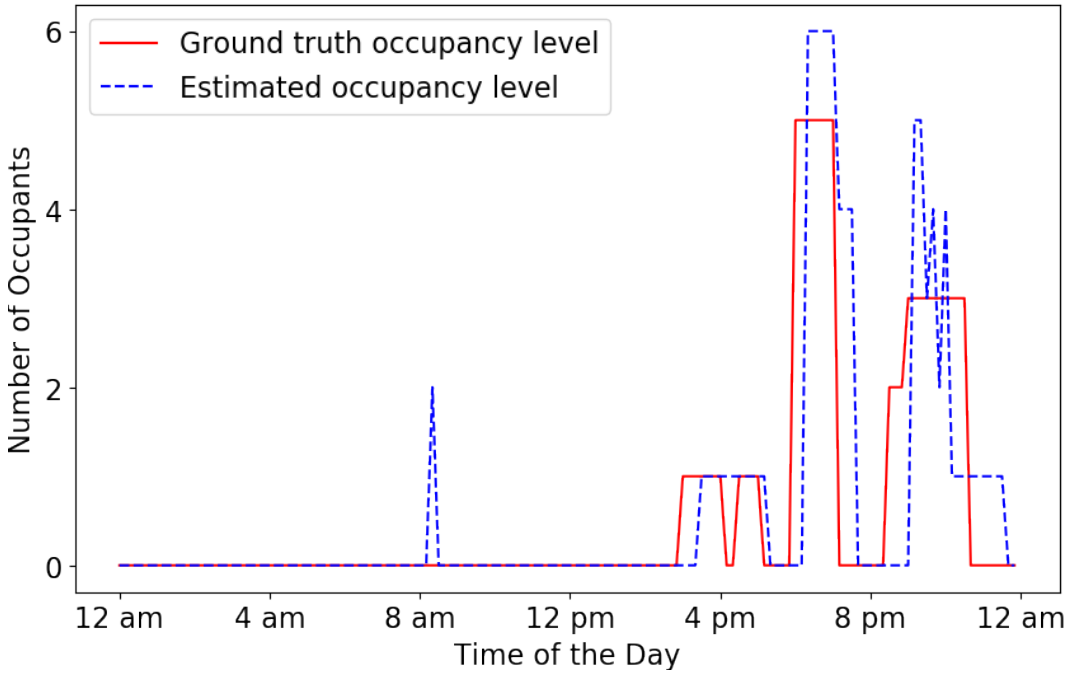


Figure 3.6: Estimated and true occupancy levels of a study area in Building A over one day.

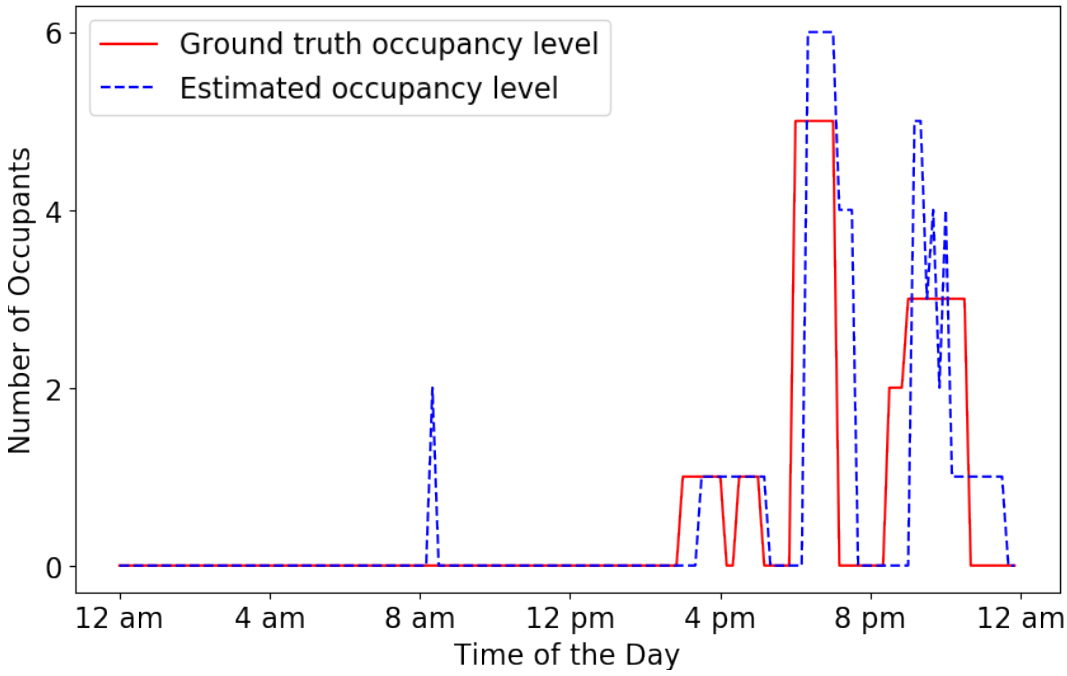
Figure 3.5 and Figure 3.6 depict the true and estimated occupancy counts in a lecture room and a study area in Building A, respectively. The lecture room reached its maximum occupancy during the day shown in Figure 3.5. The prediction is obtained from the LSTM model trained using the semi-supervised domain adaptation technique with re-weighting and retraining. We specifically use 3 hours of labelled data in the target domain to retrain the model. The data plotted in these figures is not part of the data used for retraining. Both figures suggest that the adapted model estimates the number of occupants accurately and successfully detects the overall occupancy pattern of the room. This confirms our main thesis that the number of occupants can be accurately estimated from trend data available through a rudimentary BMS.

These results not only prove the data-driven model can be used for occupancy estimation based on HVAC sensors, but also show that our proposed domain adaptation technique can increase the accuracy of the model when only a small amount of ground truth data is available in the target domain.

Figure 3.7 shows the true and estimated occupancy of a meeting room in Building B during one day, where the semi-supervised domain-adaptive LSTM



(a)



(b)

Figure 3.7: Estimated and true occupancy levels of a meeting room in Building B over one day.

model is used for occupancy estimation. Specifically, we choose Room A as the source domain and Room B which has a different size and layout as the target

domain, and assume that labelled data is available for one day in the target domain. The RMSE and nRMSE of this model are 1.87 and 18%, respectively, suggesting that the model successfully detects the overall occupancy pattern although its accuracy is lower in some intervals (for example, before and after meetings). The low accuracy in these intervals can be attributed to the fact that ground truth occupancy is extracted from the calendar, but occupants may arrive before the start of a meeting or leave before the meeting ends. We plan to investigate this in future work using a more reliable method of collecting ground truth data.

3.3.2 Changing the amount of labelled data available in the target domain

In this section, we investigate how much ground truth data is needed in the target domain to train an accurate model for occupancy estimation. Suppose the source domain is Room 3 and Room 4, and the target domain is Room 1 and Room 2 in Building A.

Table 3.3 shows the performance of supervised learning models trained using labelled data from the target domain only when it is provided for 1 hour, 3 hours, and 1 day. Although the LSTM model had the highest accuracy when only 1 hour labelled data is available, the SVR model outperforms the LSTM model when more labelled data becomes available. Therefore, we choose SVR as our supervised learning algorithm and compare the accuracy of this model with unsupervised and semi-supervised domain adaptation models. Note that we cannot train a logistic regression model when only 1 hour or 3 hour worth of labelled data is available.

Next, we build different models using supervised learning (i.e., the SVR algorithm), and unsupervised and semi-supervised domain adaptation (with re-weighting) when ground truth data is available for 1 hour, 3 hours, and 1 day in the target domain. We compare their accuracy in determining the number of occupants. As shown in Figure 3.8, the RMSE of the model developed using unsupervised domain adaptation is always the same. This is expected because the unsupervised model does not use the labelled data from the tar-

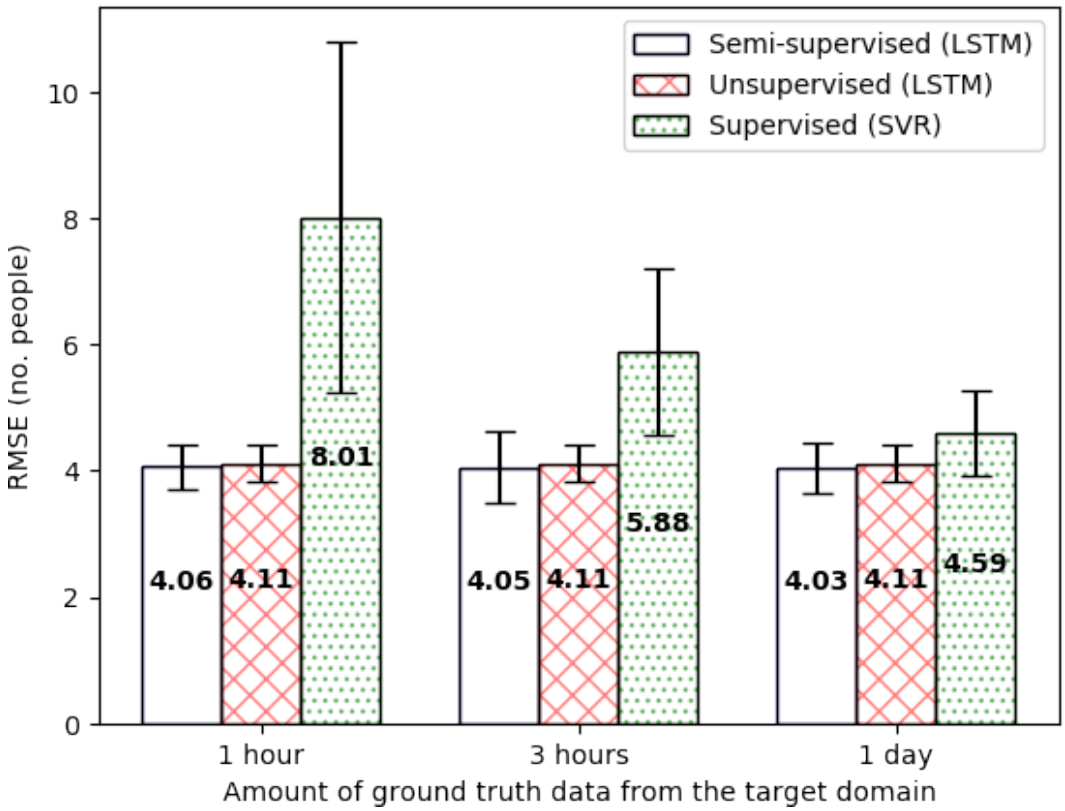


Figure 3.8: The impact of increasing the amount of available ground truth data on the accuracy of different approaches. Error bars represent the 97% confidence interval. The target domain includes Room 1 and Room 2 in Building A.

get domain. Another observation is that the accuracy of the semi-supervised domain adaptation model improves only slightly when more labelled data becomes available. This suggests that we do not need an abundance of labelled data when we use the semi-supervised domain adaptation technique and we get satisfactory performance when only 1 hour labelled data is available. Finally, it can be seen that the accuracy of the supervised learning model (SVR) improves markedly (by 40%) as more labelled data becomes available. But its accuracy is still less than that of the semi-supervised domain adaptation even when training data is available for 1 day.

Based on the results of the two buildings, we conclude that the model trained using the semi-supervised domain adaptation technique with re-weighting achieves the highest accuracy in predicting the number of occupants (an nRMSE of 17%) among other models when only a small amount of labelled data (e.g.,

1 hour) is available in the target domain. However, when sufficient labelled data is available, the model trained using SVR outperforms other models with an nRMSE of 12%.

To highlight the benefit of performing domain adaptation, we compare it with two cases where we train supervised learning models using all labelled data from the target domain, and from both source and target domains. Suppose all labelled data are available in the target domain and consider the supervised learning model (i.e., the SVR model) trained with 80% of this data from the target domain, using the other 20% for testing. The RMSE of this model is 3.72 (nRMSE of 10%), which is only slightly better than the semi-supervised domain-adaptive model trained using only 1 hour labelled data from the target domain. Furthermore, if we train a supervised learning model using all labelled data from both source and target domains without performing any adaptation, the RMSE increases to 7.87 (nRMSE of 21%). In this case training data come from two different distributions and cannot be simply combined to train a model. Using the same training data, the semi-supervised domain-adaptive LSTM yields an RMSE of 3.66 (nRMSE of 10%).

3.3.3 Different choices for source and target domains

We now change the source and target domains and compare the accuracy of different models in each case. Figure 3.9 shows the RMSE obtained for different combinations of source and target domains. In each case, the target domain is the two rooms identified below the figure, while the source domain is the other two rooms in Building A. In all cases we use 1 day labelled data from the target domain. The bar with a solid fill shows the RMSE of the semi-supervised domain adaptation method, and the bar with a hatch fill shows the RMSE of the best supervised learning method.

Note that we need to carry out re-weighting when the target domain is Room 1 and Room 2 and when it is Room 3 and Room 4 (the two leftmost pairs of bars). This is because Room 1 and Room 2 are both study areas, and Room 3 and Room 4 are both lecture rooms, causing the source and target domains to have completely different parameters: floor area, ventilation power,

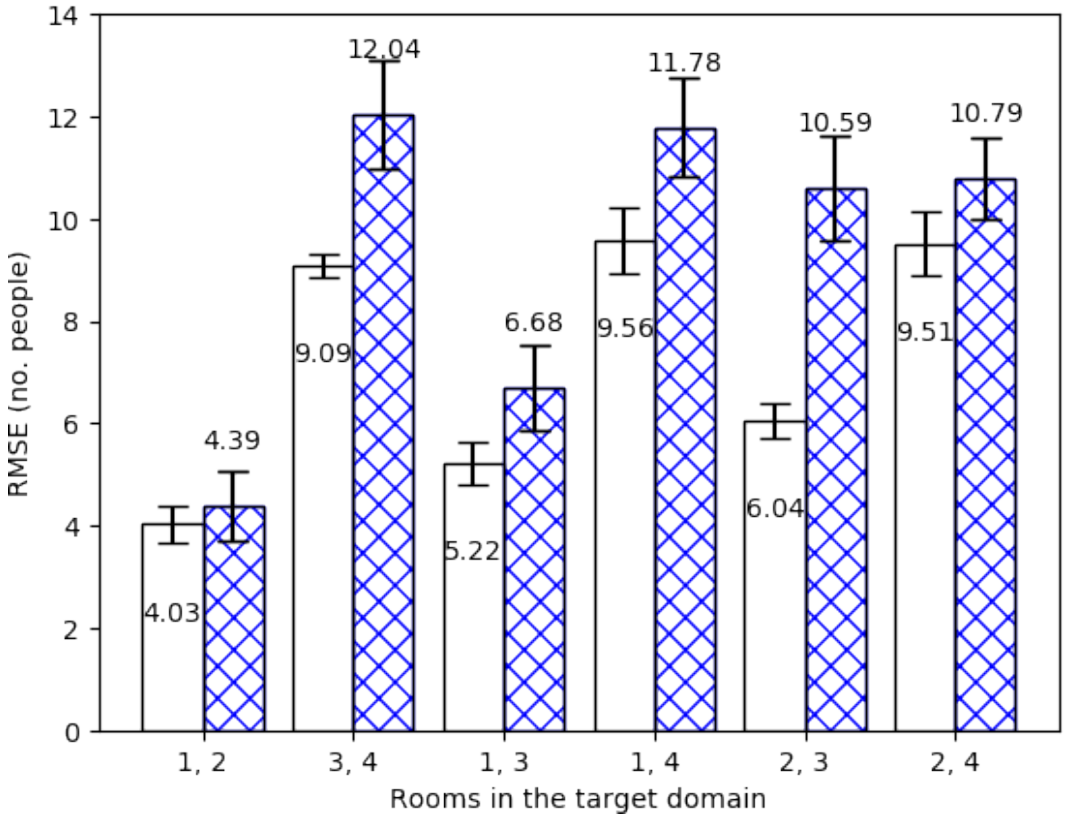


Figure 3.9: The RMSE of supervised and semi-supervised domain adaptation algorithm on different target domains given one day training data from the target domain. Error bars represent the 97% confidence interval.

etc. Without re-weighting, it is expected that the adapted model performs poorly in the occupant count determination task because the two domains are completely different. In the other four cases, both source and target domains contain one study area and one lecture room. Hence, the re-weighting step is unnecessary since the parameters in the two domains are the same. Therefore, the domain adaptation methods will not include the re-weighting method.

It can be readily seen that the average RMSE of the semi-supervised domain adaptation technique with re-weighting (i.e., the first two target domain pairs) is 6.5575, the average RMSE of the same technique without re-weighting (i.e., the last four target domain pairs) is 7.5854, and the average RMSE of the supervised learning method across all target domain pairs is 9.3785. This suggests that the domain adaptation method works better when the source and target domains have different settings and re-weighting is performed.

3.4 Discussion

To achieve scalable deployments of building applications across the building stock, it is necessary to adapt existing occupancy models to new buildings. In this chapter we showed that only 1 hour labelled occupancy data from the target domain is sufficient for the proposed semi-supervised domain adaptation technique to train a relatively accurate model for the target domain. This justifies the effort to collect a small amount of ground truth occupancy data so that building applications can adapt and reuse existing well-suited models.

Despite the novelty of our semi-supervised domain adaptation technique, it has several limitations. First, in our experiments, the source and target domains are different rooms within the same building (Building A or Building B). Thus, they are located in the same geography and share the same building envelope. This assumption makes it easier to adapt the model that is trained in the source domain. Second, the source and target domains share the same feature space, i.e., they have the same sensing modalities. Thus, it is not necessary to change the structure of our neural network model when we apply the domain adaptation technique.

In future work, we plan to investigate whether it is possible to train an occupancy detection model in one building and adapt to another building (possibly in a different climate) such that the performance of the adapted model is better than the performance of supervised learning models trained in that building using the available training data. Furthermore, we intend to study the domain adaptation problem when the two domains do not have the same feature space. Borrowing ideas from [103] and [52], which transfer the features into a latent space, we believe that this problem can be addressed in future work.

Chapter 4

Occupancy Detection Toolkit

Recent years have witnessed a steady increase in the number of occupancy detection algorithms and people counting systems designed for residential and commercial buildings, yet comparing the accuracy of existing solutions has been impossible to date due to the lack of publicly available test data sets, open-source implementation of the state-of-the-art algorithms, and consensus on the evaluation metrics. This chapter addresses this problem by presenting the design and implementation of an open-source toolkit for occupancy detection. ODToolkit is capable of importing and converting sensor data acquired from various buildings into a common data format, provides implementation of a broad suite of data-driven occupancy detection techniques, and calculates a set of evaluation metrics for each experiment. We present several case studies to show how this toolkit facilitates the development of new occupancy detection algorithms. In particular, we extend this toolkit by implementing novel domain-adaptive occupancy detection algorithms and compare them with the benchmark supervised learning algorithms on multiple data sets. Furthermore, we investigate what sensing modalities and precision are needed to achieve a desired level of accuracy for occupancy estimation through sensor fusion. ODToolkit code and documentation are available at `https://odtoolkit.github.io/`.

In this chapter, we present the design and implementation of ODToolkit– an open-source occupancy detection toolkit which is available on GitHub. ODToolkit has a modular design and can be easily extended by users. It

is comprised of five main modules for data wrangling, preprocessing, analysis, evaluation, and plotting. The first module pulls in data sets from different public repositories, such as GitHub, and converts them to a unified data format, so that they can be efficiently stored and retrieved. The preprocessing module tackles various data quality issues in a number of steps. The data analysis module allows for training and testing a broad suite of supervised learning models and semi-supervised domain-adaptive models provided in this toolkit or added by users. and performs hyper-parameter optimization for each algorithm. Users can train different occupancy models using the provided algorithms. The evaluation module includes a suite of metrics for comparing the models, and the plotting module provides various methods to create plots and showcase the results.

The rest of this chapter is organized as follows. Section 4.1 explains the pipeline and detailed information of the toolkit. Section 4.2 describes the occupancy estimation algorithms and introduce the evaluation metrics. Section 4.3 uses several case studies to demonstrate the usage of the toolkit and answer some research questions, and Section 4.4 presents discussion points and suggests directions for future work.

4.1 ODToolkit pipeline

ODToolkit is a Python package that offers open-source implementation of various occupancy detection techniques and allows for comprehensive evaluation of these models on publicly available data sets collected from residential and commercial buildings[1]. It is similar to machine learning toolkits, such as scikit-learn [87], Cloud AutoML [49], GraphLab [70], and is inspired by the design of NILMTK [16] which is a toolkit for evaluating energy disaggregation algorithms. Being written in Python, it utilizes several Python libraries, such as NumPy, pandas, SciPy, Scikit-learn, and Matplotlib, for data wrangling and cleansing, machine learning, and visualization. Figure 4.1 shows the components of ODToolkit. In the remainder of this section, we introduce

[1]ODToolkit code and documentation are available at `https://odtoolkit.github.io/`

each of these components and discuss how the design of this toolkit allows for swapping in and out features efficiently and loading new data sets.

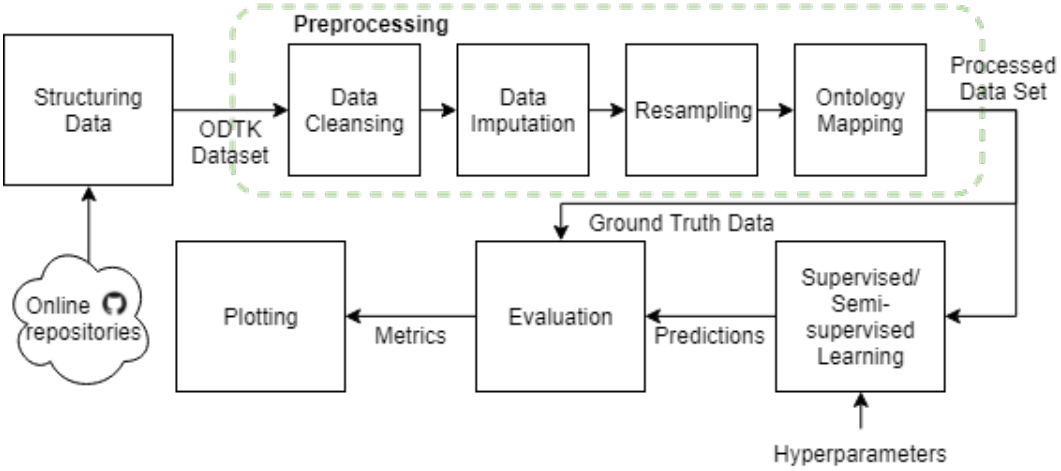


Figure 4.1: Overall architecture of ODToolkit.

4.1.1 Structuring data

ODToolkit currently contains a collection of data sets imported from public repositories and converted into a common data format. It also provides a function to load data sets, use them to train models, and compare their accuracy. This function can automatically detect new data sets (in CSV, JSON, or plain text format) that exist in a specific folder, making it easy to add new data set by just dropping them in that folder.

Public Data Sets

Five data sets are currently imported into ODToolkit and are stored using a data structure discussed in the next section. All these data sets have been previously used in related work to estimate the building occupancy level. Table 4.1 provides some statistics about these data sets and Table 4.2 shows the occupancy-indicating features in each data set.

Data Set A contains three periods of data collection in an office which lasted for five days, two days, and seven days [23]. In each period, room temperature, humidity, light level, and carbon-dioxide concentration were collected at one-minute intervals. The door of the office was mostly open in two of these

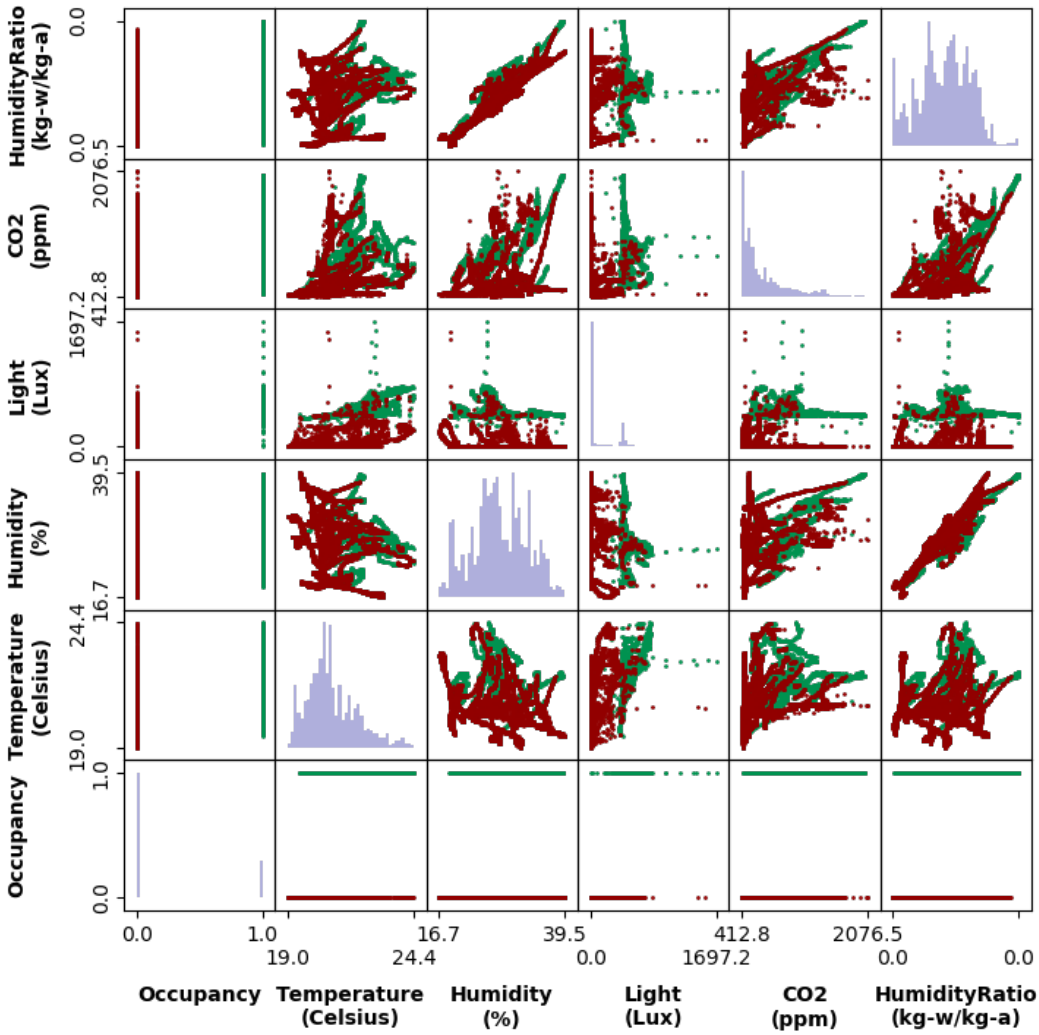


Figure 4.2: Correlation of different features in Data Set A with the two occupancy states (painted in red and green).

periods, while it was mostly closed in the third period. Binary occupancy labels were obtained from a digital camera. This data set is used in [23] to assess the performance of a small number of occupancy detection models and investigate the importance of features. Figure 4.2 depicts the correlation between different features in Data Set A.

Data Set B includes carbon dioxide and damper position data from four rooms in a campus building [6]. Data is collected in one-minute intervals from March 22, 2017 to April 5, 2017. Eight vision-based occupancy detection systems (*i.e.*, people counting cameras) are employed to count the number of occupants during the time data was collected from these rooms. This data set is previously used in [96] to detect the number of occupants using a physical model of the indoor environment, and in [106] to count the occupants using recurrent neural networks.

Data Set C contains minutely electricity consumption of two homes, for a total of three weeks in 2013 [14]. Ground truth occupancy data is obtained through the use of a tracking application running on smartphone reporting the GPS location of the individual. Reference [27] uses this data set to study non-intrusive occupancy monitoring.

Data Set D contains 116 features collected from an office building in Philadelphia, USA, between July 2012 and July 2013 at 15-minute intervals [66]. The features include longitudinal data on the thermal conditions and related behaviors. Despite the large number of features in this data set, only a fraction of them had valid data simultaneously. Binary occupancy information is also logged manually for this duration. Reference [80] uses this data set in an intrusion detection application.

Data Set E includes data from a residential building in KIT's Energy Smart Home Lab [39]. It was previously used for occupancy detection in [47]. Three types of sensors were installed in this building, monitoring the Volatile Organic Compounds (VOC) concentration, the number of Bluetooth Low Energy (BLE) key fobs in the range of a BLE receiver, and the number of connected network devices (NW) every 10 seconds. The number of occupants is logged manually for 52 days from August 10, 2016 to September 27, 2016.

Table 4.1: Summary of 5 publicly available data sets imported and analyzed by ODToolkit

Date set	A [23]	B [6]	C [14]	D [66]	E [39]
Granularity	1 min	1 min	1 min	15 min	10 sec
Occupancy label	Binary	Count	Binary	Binary	Count
Collection method	Camera	Camera	GPS	Manual	Manual
Dropout rate	0%	0%	0.14%	93.43%	0%
No. features	6	3	2	10	5
No. rooms	3	4	3	24	1
No. time slots	20560	97440	30240	35041	377549
Pct. time occupied	23.10%	45.89%	72.28%	22.82%	23.99%
Duration	14.25 days	17 days	7 days	1 years	43.66 days

Table 4.2: Features available in each data set

	A	B	C	D	E
Temperature	•			•	
Humidity	•			•	
Light	•			•	
CO2	•	•		•	
HumidityRatio	•				
DamperPosition		•			
LoadPower			•		
AirVelocity				•	
RadiantTemperature				•	
OutdoorTemperature				•	
OutdoorHumidity				•	
OutdoorAirVelocity				•	
VOC					•
Network					•
Bluetooth					•

Data format

Each data set must be transformed into a common data format before it can be efficiently processed by downstream modules in the pipeline. We refer to this data format as ODTK-Dataset. The ODTK-Dataset describes the whole data set with all rooms combined together; it stores data points in a NumPy array, and saves the names of features in a dictionary. We use NumPy multidimensional arrays because most operations we need are more efficient on this data structure compared to the list data type, and they can be easily converted to other data types, such as pandas DataFrame. Another advantage is that NumPy is a widely used library and several Python libraries rely on it. ODToolkit assigns a unique index to each feature that does not have a name.

For ease of use, ODTK-Dataset supports various ways of accessing the data set. For instance, user can give the name or the index of the room as the key to get features and occupancy labels for that specific room. In addition, ODToolkit supports two occupancy encoding methods: value encoding and

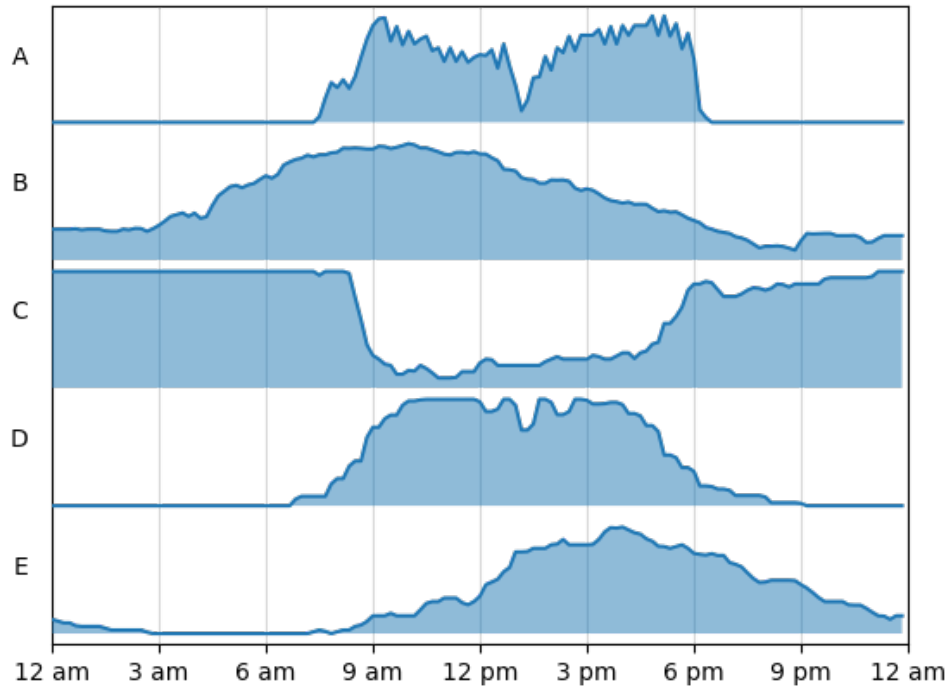


Figure 4.3: Fraction of time each building is occupied at any given time of the day.

one-hot encoding. The ODTK-Dataset retains information about the names of the features, size, function, and name of rooms (if provided), the occupancy label, and corresponding features. It allows for including or excluding certain features from a data set. Thus, users are capable of using ODTK-Dataset to run all occupancy detection algorithms provided as part of this toolkit.

ODToolkit allows users to import and export ODTK-Datasets at any time. An ODTK-Dataset can be dumped into the disk as a binary file or a JSON file, and can be loaded back when needed. Everything in the data set folder must be saved by this function as a binary file. The read/write speed of this binary file is on average 20 times faster than the original data file.

4.1.2 Collecting statistics about data sets

Occupancy data sets are collected from different types of buildings instrumented by different types of sensors. There is also a variety of approaches to acquiring and storing sensor data. Hence, each ODTK-Dataset has unique elements. To understand these elements, inform the transformation and analysis

processes about missing or low quality data, and investigate how they could impact the accuracy of occupancy detection models, ODToolkit provides functions to analyze an ODTK-Dataset and collect certain statistics about the entire data set, a particular room in the data set, or a particular feature in that room (or in all rooms in that data set). We describe these statistics below:

- **Frequency** indicates the average sampling frequency of all features in an ODTK-Dataset.

- **Dropout rate** indicates the percentage of data points missing in an ODTK-Dataset. Concretely, it is defined as

$$Dropout = \frac{\text{\# of valid data points} \times \text{average frequency}}{\text{time duration}}$$

- **Gap** is defined as the case where the difference between time-stamps of two successive values exceeds a predefined threshold.

- **Occupancy distribution over time** is the distribution of different occupancy states in an ODTK-Dataset. The knowledge of this distribution is essential when interpreting the accuracy of an arbitrary model. For example, 95% accuracy may not be acceptable if it is achieved in a room which is always unoccupied. Figure 4.3 shows the percentage of time each building was occupied over the course of the day. If there are more than one room in a building, we plot the average of their occupancy distributions.

- **Uptime** is the length of time a sensor reported values. It differs from the dropout rate in that it captures the length of the time interval in which data points were available.

For all analysis functions above, ODToolkit has two options on data sets: data-set-based and room-based. For data-set-based function, the results are based on the whole data set. For example: in data-set-based occupancy evaluation, only the distribution cross all rooms is returned. However, in room-based occupancy evaluation, the function will return multiple distributions for each room.

Also, ODToolkit has two options on features: data-set-based and sensor-based. For example, in data-set-based gap detection function, if a data point has any feature with an empty value, then the data point is invalid. However, in sensor-based gap detection function, each feature is independent, so there may have some gaps on only one feature.

4.1.3 Pre-processing

Sensor data collected from the built environment are often dirty, containing noisy, erroneous, and missing values. This data must be cleaned before being used by machine learning algorithms. ODToolkit provides various functions to clean and normalize data, detect and remove outliers, impute missing values, resample at a desired frequency, and change the feature representation. We discuss some of these functions below:

Outlier detection: ODToolkit could adopt different outlier detection algorithms. The default algorithm requires calculating the interquartile range (IQR), *i.e.*, the difference between the largest and smallest values in the middle 50% of the data set. ODToolkit then flags any value more than 1.5 IQR above the third quartile or below the first quartile as an outlier[2]. Outliers and missing values are treated similarly.

Missing data imputation: ODToolkit uses a forward-filling method to impute missing data. More specifically, it uses the last value before a gap to fill in the gap. When forward-filling is not feasible, ODToolkit adopts backward-filling to tackle missing data. More sophisticated algorithms, such as multiple imputation, can be easily added to the toolkit.

Changing the sampling frequency: Building occupancy data sets contain measurements from various sensors with distinct sampling rates and precision. This leads to the problem of having non-uniform sampling intervals which complicates multivariate analysis of this data and training models. ODToolkit tackles this problem by changing the sampling frequency of different data streams in ODTK-Dataset; it has two functions for upsampling and down-

[2]The threshold used for outlier detection can be modified by user depending on the distribution of data.

sampling. Downsampling is the process of increasing the sampling frequency of data. It involves computing the mean of multiple data points in a fixed size window and replacing them with the mean. In contrast, upsampling is the process of decreasing the sampling frequency and requires performing some sort of interpolation. ODToolkit uses linear interpolation by default.

Encoding ground truth occupancy data: ODToolkit can change the encoding of occupancy data in ODTK-Dataset from the actual number of occupants to the binary occupancy state. Moreover, it can carry out one-hot encoding, when necessary.

Naming convention: ODToolkit identifies features representing the same physical quantity across all data sets, and unifies their names. Inspired by [13], [23], [72], ODToolkit assigns a name that consists of two parts, *i.e.*, a location identifier and a *tag*, to each feature. The location identifier can be INDOOR, OUTDOOR, and UNKNOWN. Tags provide a consistent way of annotating features associated with the same physical quantity and enable users to run models that expect a specific feature on multiple data sets containing this feature, yet with different names. For example, we use the CO2 tag for both CO2Meter and CO2GasSensor features. ODToolkit comes with a vocabulary of acceptable tags which may be extended by user. For any new feature found in a data set, ODToolkit computes the similarity between its name and tags in its vocabulary. If it is sufficiently similar to an existing tag (using a predefined threshold), ODToolkit annotates the feature using this tag. Otherwise, it extends its vocabulary by defining a new tag.

To measure the similarity between a feature name and a tag, ODToolkit computes the Jaro similarity of two strings. Jaro similarity is an accurate metric for string comparison and can be computed efficiently [91]. It gives a more reliable result than Hamming distance and is computed faster than Levenstein distance. The higher the Jaro distance is, the more similar the two strings are. The Jaro distance is normalized such that 1 indicates an exact match and 0 indicates no similarity between the strings. It is also close to one

when one string is a prefix of another string. It is defined as:

$$Jaro(a,b) = \begin{cases} 0 & \text{if } m = 0, \\ \frac{1}{3}\left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m}\right) & \text{otherwise,} \end{cases}$$

where m is the number of matching characters, and t is half the number of transpositions. Two characters from a and b are called matching characters if they are the same and their indices are no more than $\lfloor \frac{\max(|a|,|b|)}{2} \rfloor - 1$ apart.

4.2 Methodology

ODToolkit provides the implementation of several data-driven occupancy detection models. Data-driven models are trained using all features that are available in a data set without taking into account the physical properties of each room. Note that some of these models can only be used for binary occupancy detection and are not suitable for occupant count determination. In addition, ODToolkit provides various evaluation metrics for both binary occupancy detection and occupant count determination tasks.

4.2.1 Supervised learning models

Several supervised learning models proposed in related work are added to ODToolkit. These include standard black-box models, such as Random Forest (RF), Hidden Markov Model (HMM), Support Vector Machine (SVM), Artificial Neural Network (NN), Recurrent Neural Network (RNN), and Sparse Non-negative Matrix Factorization (SNMF), which are introduced in Section 2.1.

For supervised learning models, ODToolkit requires two data sets: training data set and testing data set. The training data set must include all ground truth occupancy labels, and the two data sets must be collected from the same environment.

4.2.2 Evaluation metrics

ODToolkit contains 16 standard metrics to evaluate the performance of various occupancy detection models. This set can be easily extended by adding the definition of a new metric to the evaluation folder. This way the new metric

is automatically picked up by ODToolkit. Suppose the predicted occupancy state is denoted by $\hat{y}_t$ at t and the true occupancy state is denoted by y_t. We describe these metrics below:

True positive (TP), False positive (FP), True negative (TN), and False negative (FN) are used to evaluate binary occupancy prediction models. The TP rate indicates the proportion of the actual occupied states that are correctly identified as such, the FP rate indicates the proportion of the actual unoccupied states that are identified as occupied, the TN indicates the proportion of the actual unoccupied states that are correctly identified as such, and the FN indicates the proportion of the actual occupied states that are identified as unoccupied.

Precision, Recall, Fall-out, Miss-rate, Selectivity are used to evaluate the performance of binary occupancy detection models. The two main metrics are precision and recall. Precision indicates the percentage of correct occupancy predictions by:

$$Precision = \frac{TP}{TP + FP}$$

While recall is the percentage of the true occupied states:

$$Recall = \frac{TP}{TP + FN}$$

Fall-out is defined as

$$Fallout = \frac{FP}{FP + TN}$$

Miss-rate:

$$Missrate = \frac{FP}{FP + TN}$$

And selectivity:

$$Selectivity = \frac{TP}{TP + FN}$$

F1-score is the harmonic average of the precision and recall, and can only be used for evaluating binary occupancy prediction models. It is defined as

$$F_1 = \frac{2 \times TP}{2 \times TP + FP + FN}$$

Accuracy is the only evaluation metric which can be used to evaluate the performance of binary occupancy detection and occupant count prediction models. ODToolkit allows users to set a tolerance for this metric. If the difference between predict occupancy level and the actual occupancy level is less than this parameter, ODToolkit considers the prediction correct.

$$Accuracy = \frac{1}{T}\sum_{t \in T} count_t$$

$$count_t = \begin{cases} 0, & \text{if only one of } \hat{y}_t \text{ and } y_t \text{ is 0} \\ 0, & \text{if } |\hat{y}_t - y_t| > \tau \\ 1, & otherwise \end{cases}$$

The parameter τ is set to zero in the binary occupancy prediction task, while it can take any positive value (defined by the user) in the occupant count determination task.

Root Mean Square Error (RMSE) is a measure of the differences between predicted and observed occupancy states. We use RMSE to evaluate models that determine the number of occupants. Smaller RMSE values indicate more precise predictions:

$$RMSE = \sqrt{\sum_{t \in T} (\hat{y}_t - y_t)^2}$$

Normalized RMSE (nRMSE) is similar to RMSE but gives a relative sense of the accuracy by normalizing it by the difference between the minimum and maximum of the true occupancy count[3]:

$$nRMSE = \frac{\sqrt{\sum_{t \in T} (\hat{y}_t - y_t)^2}}{\max y - \min y}$$

Mean Absolute Error (MAE) is the average magnitude of the absolute errors. Unlike RMSE which puts a heavier penalty on large errors, MAE penalizes error values similarly.

$$MAE = \frac{1}{T}\sum_{t \in T} |\hat{y}_t - y_t|$$

[3]ODToolkit also supports another definition of nRMSE, which normalizes based on the average of the true occupancy count.

Mean Absolute Percentage Error (MAPE) is a measure of relative accuracy similar to nRMSE. However, it is not symmetric and cannot be used when the true occupancy state is zero since it results in a division by zero.

$$MAPE = \frac{100\%}{T} \sum_{t \in T} \left| \frac{\hat{y}_t - y_t}{y_t} \right|$$

Mean Absolute Scaled Error (MASE) is a scale invariant measure of prediction accuracy that penalizes positive and negative errors equally. In particular, it normalizes the absolute prediction error with respect to a baseline that uses the previous value as a predictor of the next value.

$$MASE = \frac{\sum_{t=1}^{T} |y_t - \hat{y}_t|}{\frac{T}{T-1} \sum_{t=2}^{T} |y_t - y_{t-1}|}$$

4.3 Case studies

We study the usability of ODToolkit through several case studies. In particular, we discuss how the toolkit can be extended with new algorithms and how it facilitates comprehensive evaluation of various occupancy detection models, including the models developed in this work and the models proposed in related work and implemented in the toolkit across residential and commercial buildings. In particular, we use this toolkit to (a) investigate the importance of different sensing modalities for occupancy detection, (b) understand how different models that rely on the same set of features perform on a given building and how their differences can be explained, (c) reason about why a given occupancy detection model performs poorly on one building, while it works well on other buildings, (d) find out if low-cost, noisy sensors can provide input to occupancy detection models, (e) explore if the models originally developed for binary occupancy detection can be used to discern the number of occupants, and (f) compare the accuracy of domain-adaptive and supervised learning models in various occupancy detection tasks and discuss how much ground truth data is necessary from the target domain to train a domain-adaptive model with a sufficient accuracy level.

In the following, we address the above questions and show how ODToolkit helps researchers gain key insights about occupancy detection models and data

sets. To save space, we choose a single data set to discuss each case study below, but all these experiments can be repeated on other data sets using the toolkit.

4.3.1 Extending the toolkit

The first case study involves adding a new class of occupancy detection models (i.e., domain-adaptive models) to the toolkit. Unlike the supervised learning models that are trained and tested in the same room, the domain-adaptive models are pre-trained using the abundance of occupancy labels that are available from one room in a data set and are tested after some adaptation in another room, within the same data set or in another data set, where occupancy labels are presumably sparse or nonexistent. The former room is referred to as the *source domain*, while the latter is referred to as the *target domain*. We first present these models and then discuss the usability of the toolkit.

Domain-adaptive models

Three domain-adaptive models, namely domain-adaptive hidden Markov model (DAHMM) domain-adaptive particle filter (DAPF), and domain-adaptive long short-term memory (DALSTM), are added to ODToolkit as part of this case study. DAHMM and DAPF is proposed in [105], which is not part of the thesis, but implemented in ODToolkit. All models are pre-trained in the source domain using the algorithms presented in Section 2.3.

Domain-adaptive Long Short-Term Memory (DALSTM): DALSTM is an extended version of LSTM and relies on the LSTM model trained in the source domain [106]. The algorithm is same as the work in Section 3.2. It has two steps before estimating occupancy in the target domain: a re-weighting step followed by a re-training step. In the re-weighting step, selected weights of each gate in the LSTM network are adjusted based on the known (or apparent) differences between the source and target domains [106]. If physical properties of the two domains are known, the weights of the input nodes for all gates are adjusted. Similarly, re-weighting should happen for the output layer. The distribution of two domains becomes closer after re-weighting because of

the mapping of weights for each features. After re-weighting, we use the available labels from the target domain to re-train the LSTM model. After these two steps, the adapted model is used for occupancy detection in the target domain.

Usability of the toolkit

The first two algorithms described above were implemented by a researcher who was not initially involved in developing the toolkit but read its documentation. The third algorithm was implemented by a researcher who developed the toolkit and was familiar with its codebase. Both researchers were asked to implement a super class for the new class of occupancy detection models and derive the domain-adaptive model(s) from this class. We interviewed them after they finished the assigned tasks to understand how they evaluate the usefulness of the toolkit. We found that using the toolkit reduced the effort and time required to build a new occupancy detection model for both of them because of three reasons:

- they used a supervised learning algorithm that was already implemented in the toolkit for pre-training the model. Hence, they only had to implement the adaptation algorithm. Additionally, they used machine learning libraries in Python, such as Scikit-learn, to build the models;
- since all data sets are transformed into the same format by the toolkit, it was easy to tackle the case that the source and target domains were in two different data sets;
- they could easily compare the new models against the benchmark models using the available metrics to verify their implementation.

We aim to evaluate the usability of the toolkit at a larger scale and in a more systematic way in future work.

4.3.2 Examining the feature importance

While there is a plethora of sensors installed in different types of buildings, not all of them are measuring quantities that are correlated with occupancy,

and include irrelevant features may confuse the model, therefore reduce the accuracy of occupancy estimation. Thus, a fundamental question is how to rank and find the most appropriate sensing modalities based on the amount of information they provide for occupancy detection. In this case study, we explore whether ODToolkit allows for investigating this question. Knowing the relative importance of features could help us build parsimonious models relying on the most distinguishing features, thereby reducing the complexity of model training.

ODToolkit provides methods for adding and removing specific features to and from a data set and evaluating how it affects the performance of models trained on that data set. To illustrate this we train an RF model on Data Set A and apply it to estimate the occupancy state. Table 4.3 shows F1-score of the RF model when we use different subsets of features. A lower score appearing on the right column suggests that the feature has more influence on predictions. A higher score appearing on the left column indicates a strong correlation between the feature and the occupancy state. The model is trained using 80% of data in Data Set A.

If we rank the most distinguishing features with respect to ANOVA F-values, we get the following order:

$$Light > Temperature > CO_2 > HumidityRatio > Humidity$$

which is consistent with the results shown in Table 4.3 and the conclusion made in [23] about the importance of features in this data set for occupancy detection.

4.3.3 Evaluating supervised learning models

The primary benefit offered by ODToolkit is the ability to compare the performance of an arbitrary model with benchmark models on different data sets. To illustrate this, we predict occupancy using supervised learning models and evaluate their performance on all five data sets.

Figure 4.4 depicts the evaluation results of supervised learning models presented in Section 2.1. All models are trained on Data Set A using 80% of data

Table 4.3: Evaluating performance of the RF model (Accuracy / F1 Score) with different features on Data Set A.

	including this feature only	**excluding this feature**
Temperature	0.8745 / 0.4901	0.9968 / 0.9889
Light	0.9966 / 0.9880	0.9270 / 0.7099
CO_2	0.8818 / 0.4927	0.9910 / 0.9677
Humidity	0.8057 / 0.2354	0.9951 / 0.9828
HumidityRatio	0.6671 / 0.3319	0.9954 / 0.9837

Table 4.4: Comparing the performance of supervised learning models across different data sets.

		Data set				
		A	**B**	**C**	**D**	**E**
RF	**Accuracy**	0.9959	0.6416	0.7624	0.8234	0.9088
	F1 Score	0.9854	0.6608	0.8364	0.6151	0.7399
NN	**Accuracy**	0.9981	0.6172	0.8884	0.8000	0.9662
	F1 Score	0.9932	0.7141	0.9279	0.1957	0.8945
LSTM	**Accuracy**	0.9981	0.6719	0.8472	0.8068	0.9412
	F1 Score	0.9932	0.7478	0.8816	0.1436	0.8947

and are applied to estimate the binary occupancy state. It can be readily seen that all data-driven models achieve a high level of accuracy in the binary occupancy detection task. Specifically, NN and LSTM obtain the highest accuracy of 0.9981, followed by SVM which achieves the accuracy of 0.6729. while all other models all have an accuracy of above 0.99. The lowest accuracy is for SNMF which is around 0.9229. These two models achieve the F1-score of 0.6489 and 0.6428, respectively.

Table 4.4 summarizes the results obtained on the five data sets using RF, NN, and LSTM models to perform binary occupancy estimation. According to this table, there is at least one model that yields above 90% accuracy in Data Sets A, C, and E. We also observe that the high dropout rate of Data Set D negatively affects the performance of all models.

Comparing the results of RF and NN across data sets, RF performs better

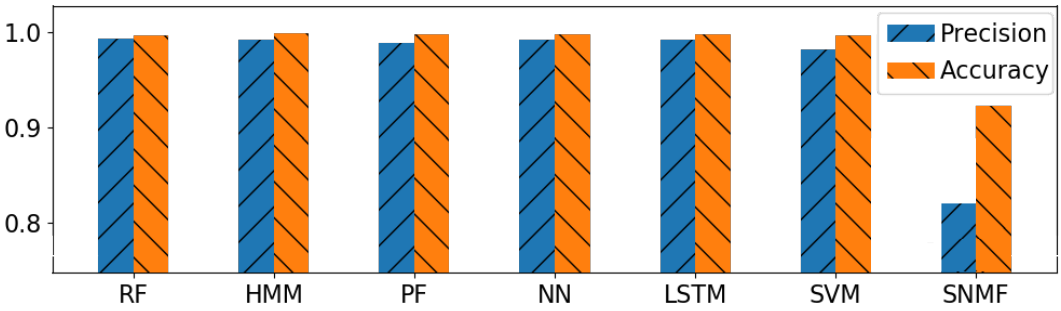


Figure 4.4: Comparing supervised learning models for binary occupancy detection in Data Set A. Note that the y-axis does not start from 0.

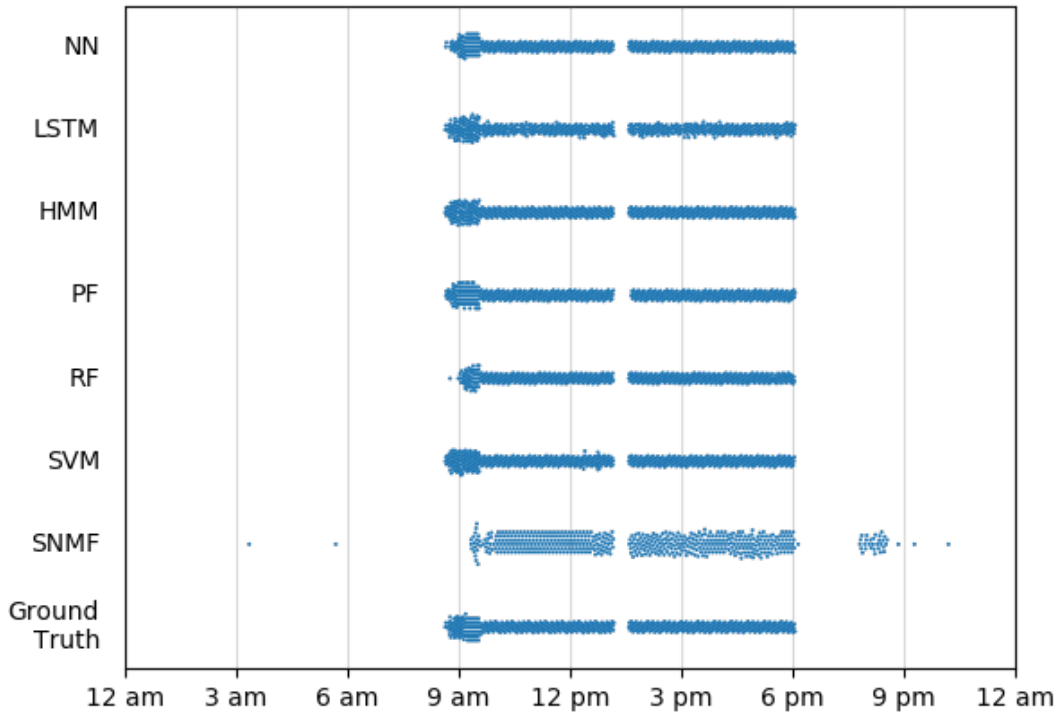


Figure 4.5: A swarm plot showing the distributions of occupancy events estimated by different supervised learning models along with the true occupancy events in Data Set A.

on Data Set D, while NN gets a higher score on other data sets. Recall that ODToolkit fills in missing values using forward-filling. This leads to having the same values for many time slots when the dropout rate is high, lowering its importance in RF. RF assigns weights to features based on their importance. Thus, it uses features with higher dropout rate less in estimation. As a result, RF performs better when data set has a higher drop out rate.

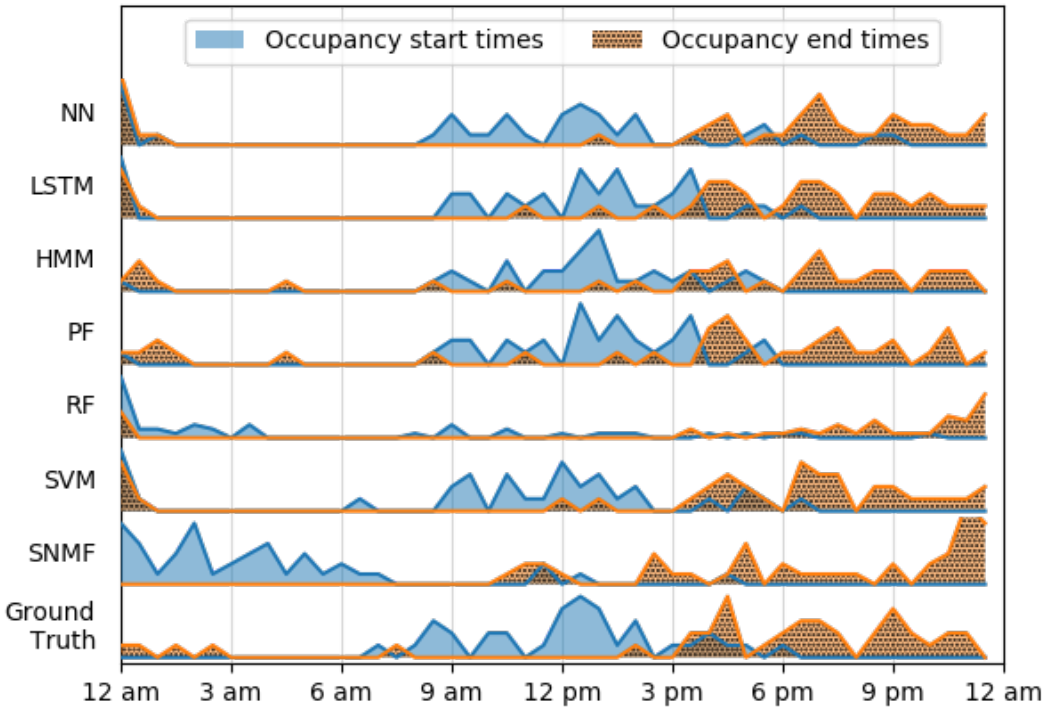


Figure 4.6: Distributions of the start times and the end times of occupancy intervals obtained by different supervised learning models in Data Set E.

Note that Data Set B contains two types of rooms (large classrooms and small study rooms) and training the same model for both types of rooms reduces the test accuracy as seen in Table 4.4. ODToolkit also supports training and testing models separately in each room. In this case, LSTM achieves F1-score of 0.85 and 0.75 for large and small rooms in Data Set B, respectively.

Figure 4.4 compares the performance of all supervised learning models on Data Set A in terms of their prediction accuracy and precision. For each model, we use 80% of data for training and report the test accuracy on the other 20%. It can be readily seen that all these models achieve high accuracy in the binary occupancy detection task. Specifically, NN and LSTM obtain the highest accuracy of 0.998, while all other models reach above 0.99 accuracy. The lowest accuracy is for SNMF which is around 0.923. Figure 4.5 shows the number of times a room in Data Set A was occupied according to the ground truth labels and estimations of different supervised learning models. We witness that SNMF has several spurious detections around 9pm.

Table 4.5: Comparing the performance of supervised learning models in binary occupancy detection and occupancy count determination tasks on Data Set B.

	Prediction Task (F1 Score/Accuracy)	
	Count	**Binary**
RF	0.6426 / 0.3825	0.6615 / 0.6421
LSTM	0.6773 / 0.4059	0.6507 / 0.6669
NN	0.2682 / 0.4132	0.7106 / 0.6288

Comparing the distribution of ground truth occupancy times with the inferred occupancy times indicates that all other models can estimate the room-level occupancy state quite accurately. Nevertheless, most models exhibit a small delay in estimating the occupancy states, which could be attributed to the fact that carbon dioxide builds up slowly.

We note that accuracy may not be the best metric to compare two occupancy detection algorithms, especially when estimated occupancy schedules will be incorporated in the HVAC control loop to minimize energy use, while maintaining occupant comfort. In particular, a practical approach to HVAC control is to condition rooms for the entire occupancy period over a day, even if they are unoccupied for a short period of time in between, for example during the lunch time [4], [102]. Considering this application, we should compare different occupancy detection models in terms of their ability to accurately predict the earliest time a room gets occupied (aka occupancy start time) and the latest time it remains occupied (aka occupancy end time) during the day. Figure 4.6 depicts the distributions of start times and end times in Data Set E using different occupancy detection models as well as the true distributions. In each case we use 80% of labelled data for model training. It can be readily seen that SNMF and RF fail to correctly identify the distribution of start times, whereas all other models have a relatively reliable estimation. RF also has high error when it comes to predicting the end times, making it the worst choice for detecting the daily occupancy schedule in this data set.

Finally, we use the toolkit to compare the accuracy of selected models in binary occupancy estimation and occupant count determination tasks. All

models are trained using 80% of data from Data Set B. As shown in Table 4.5, every model yields a higher accuracy for binary occupancy estimation than occupant count determination. This is expected as counting the number of occupants is a more difficult task and enough labels may not even exist in the training data for each occupancy level.

4.3.4 Comparing domain-adaptive models and supervised learning models

In this case study, we use the toolkit to investigate two important questions about the new domain-adaptive models: whether they outperform supervised learning models in the binary occupancy detection task and how much training data must be available in the target domain so that we get a meaningful improvement in the accuracy of the domain-adaptive models compared to supervised leaning models trained using labels from the target domain only. ODToolkit allows for changing the amount of ground truth occupancy data (labelled data) in the source domain to initially train a model and the amount of ground truth occupancy data from the target domain used for adapting the pre-trained model.

Figure 4.7 shows the comparison of F1-scores obtained by domain-adaptive and supervised-learning models which are trained in the target domain only, when we use Room 3 in Data Set A as the source domain and Room 1 in that data set as the target domain. We use all data in the source domain to train the domain-adaptive models, and vary the percentage of data in the target domain which is used to re-train the domain-adaptive models and train the supervised-learning models. Specifically, we increase the amount of training data from the target domain from 10% (4.5 hours) to 30% (13.5 hours) of all available data. It can be seen that domain-adaptive models achieve sufficiently high accuracy even when a limited amount of training data is available. Supervised-learning models require much more training data to achieve the same level of accuracy. Also all supervised learning models become more accurate as more training data becomes available from the target domain. Specifically, when only 20% of training data is available in the target domain, the F1-score of LSTM improves

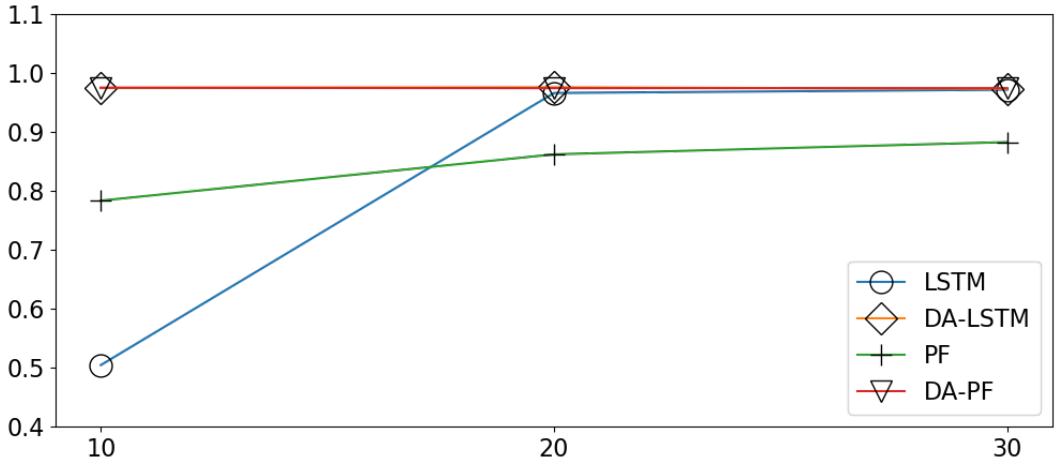


Figure 4.7: The F1-score of domain-adaptive models and supervised learning models on Data Set A using different amounts of training data form the target domain.

from 0.504 to 0.975, and the F1-score for PF improves from 0.783 to 0.975. We do not observe much improvement for domain-adaptive models because they achieve high accuracy (F1-score of 0.95 for both DALSTM and DAPF) with smaller amounts of training data.

4.3.5 Investigating model sensitivity to noise

We now study how the accuracy of a model changes when the data used to train the model is perturbed by noise. When adding noise to data points, we make sure that they are not detected as outliers and fixed by the outlier detection module of ODToolkit.

In particular, we consider Data Set A in this section and apply an additive white noise to all features in this data set, except for the occupancy label and time-stamp. In each case, the standard deviation of noise is set to one fourth of the standard deviation of the corresponding feature. This gives a reasonable level of noise and preserves the original distribution of the feature (see Figure 4.8 compare with Figure 4.2 and Figure 4.9). Figure 4.9 indicates that if the white noise has even higher standard deviation, data lose its trend and noise lead the distribution.

We consider 7 supervised learning models, namely RF, HMM, PF, NN,

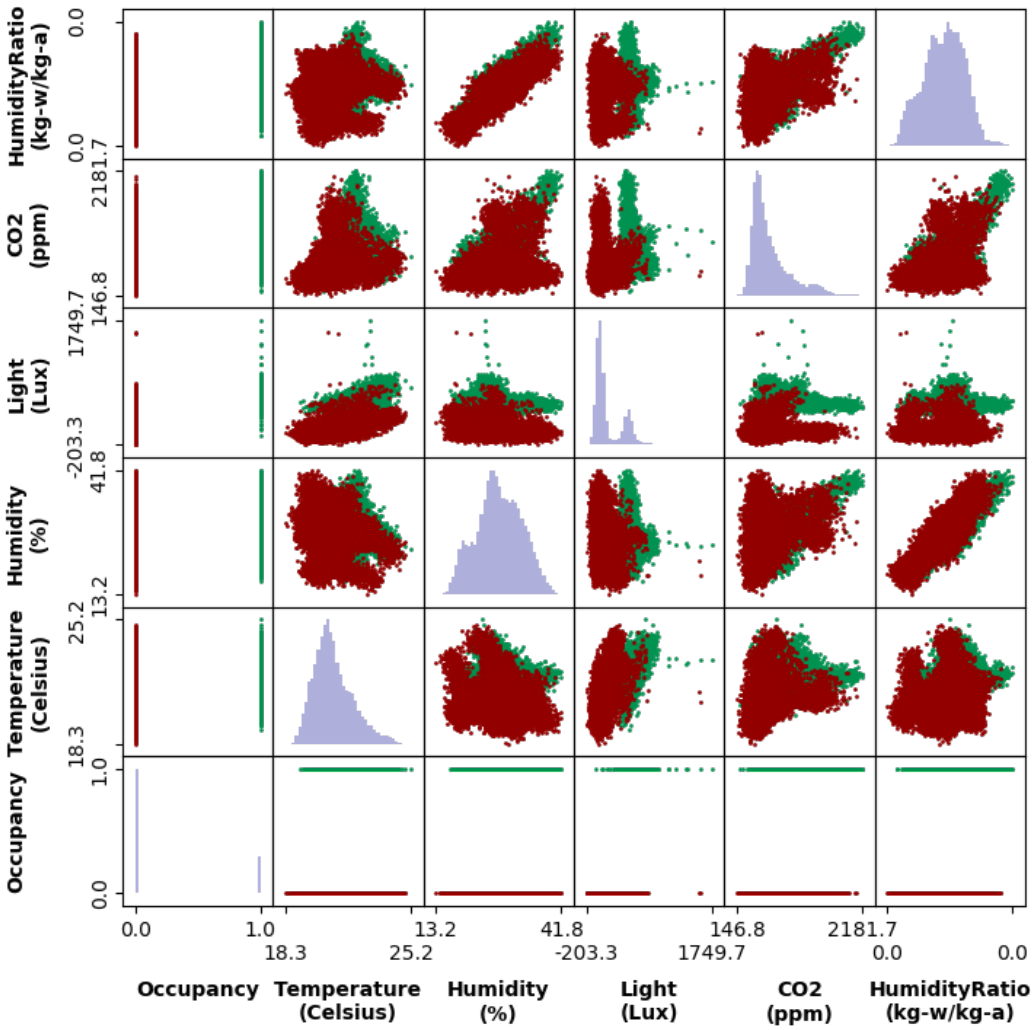


Figure 4.8: Correlation of different features in Data Set A with the two occupancy states (painted in red and green) after adding noise to the features. The main diagonal shows the probability density function of each feature.

SVM, LSTM, and SNMF, and train them using 80% of data. As a result of adding noise, the F1-score of the first 5 models decreases by about 0.01, from 0.99 to 0.98. However, the performance of LSTM and SNMF reduces dramatically, by around 0.19 (from 0.99 to 0.80) and 0.25 (from 0.69 to 0.44), respectively. We attribute this to the fact that LSTM is a recurrent neural network model with memory, and therefore, the error accumulates over time and has a more pronounced effect on the performance of the model compared to the memory-less models. In case of the SNMF model, it considers multiple rows of the matrix together as input. Thus, errors in multiple rows affect the

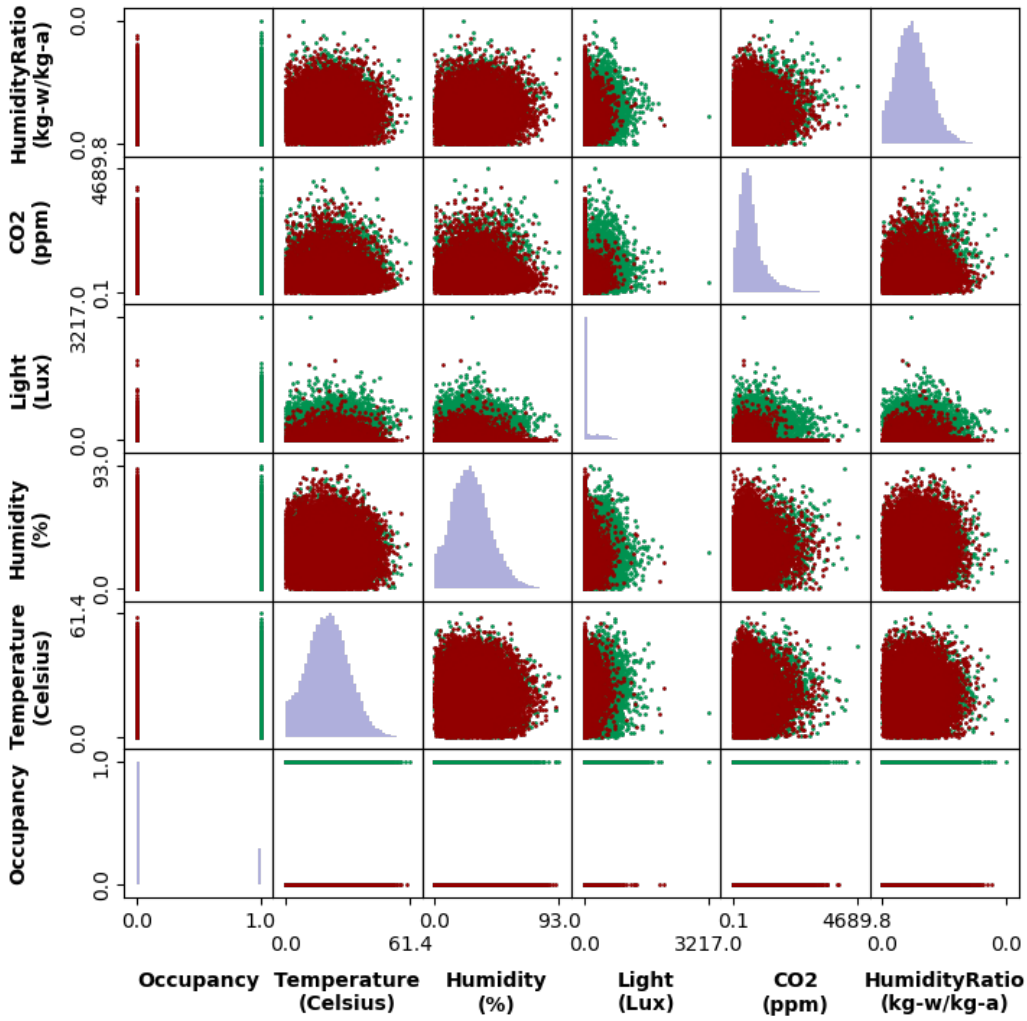


Figure 4.9: Correlation of different features in Data Set A with the two occupancy states (painted in red and green) after adding too much noise to the features. The main diagonal shows the probability density function of each feature.

SNMF's predictions to a greater extent.

Overall, our results imply that this level of noise, which can be possibly introduced by analog sensors, does not noticeably impact the occupancy detection accuracy. ODToolkit makes it possible to repeat this study for other types of noise to understand what combination of low-accuracy sensors can be used for occupancy detection given a target level of accuracy.

To understand why adding noise decreases the accuracy of RF, HMM, PF, NN, and SVM models only slightly, we use ODToolkit to plot the correlation

Table 4.6: The effect of increasing training data of a neural network on its test accuracy in Data Set A.

	Percentage of labelled Data (%)			
	20	**40**	**60**	**80**
F1 Score	0.9187	0.9369	0.9469	0.9932
Accuracy	0.9656	0.9706	0.9764	0.9981

between different features before and after applying the white noise. For a high performance before applying noise, the correlation between each feature shows in Figure 4.2. Figure 4.2 shows two identifiable cluster represents occupied state and unoccupied state. Figure 4.8 shows how features are correlated after adding noise. Comparing it with the original correlations (depicted in Figure 4.2 in the appendix), it can be seen that the two classes, viz. occupied and unoccupied, are still distinguishable. Hence, the prediction accuracy does not decrease substantially for these models which perform a simple classification.

4.3.6 Percentage of training data affects model performance

For the question of how sensitive is the performance of a given model to the amount of ground truth data available for training, ODToolkit can modify the dividing point on all data sets.

Following the property of the data-driven model, the more distinct feature data and the corresponding label is given, the more accurate the model is. The over-fitting problem rarely arises when increasing the number of distinct feature data. Therefore, a higher percentage of training data results in a better model.

The result of a simple experiment is shown in Table 4.6. We select Data Set A to train and test the NN model. The trend in Table 4.6 on both metrics strongly supports the statement of the training percentage have a positive relationship with the prediction result.

4.4 Discussion

Buildings are historically operated based on a rough estimate of their occupancy schedule, a practice that has led to significant energy waste and occupant discomfort across the building stock. With recent advances in sensor networking and data mining, it has become possible to obtain fine-grained and accurate occupancy information in an inexpensive and non-intrusive way using data-driven occupancy detection techniques. However, none of the techniques proposed in related work is validated on a large set of buildings with distinct occupancy patterns in different climates. Furthermore, it is currently virtually impossible to answer the following questions: which occupancy detection model has superior performance? how much training data does it need? is it possible to reuse models developed in a building to accurately detect the occupancy of a similar building? and what combination of sensors should be deployed in a building to increase the accuracy of the occupancy detection models? In this chapter we introduced an open-source toolkit which improves reproducibility of occupancy detection experiments and enables the research community to address the above questions. To evaluate the usability of this toolkit, we implement three semi-supervised domain-adaptive occupancy detection models and compare their performance on specific data sets using the standard evaluation metrics provided in the toolkit.

ODToolkit aims to set up an open-source platform for the occupancy detection community. With the purpose of enabling researchers to quickly evaluate their model with any available models and share their experimental resources with the community, the toolkit must easy to use, implement, as well as extend. ODTK-Dataset is the core part of the toolkit. This data structure offers a standardized data structure for the occupancy detection community. The structure provides a various way to extract, edit, and concatenate data. Also, the structure has a self-adaptive ability. It can narrow down or expand the feature set to fulfill the requirement of easy to add any new room into the data set and keep the consistency of the data set at the same time.

Besides, the soul and the reason of the community demand ODToolkit is

the expandability of the toolkit. To minimize the coding structure researcher need to understand and follow, ODToolkit uses a subclass to build models. The researcher only needs to copy two function name and parameters, and no other structure need to know. For data sets, after the researcher use functions in ODToolkit to transform the raw data set into ODTK-Dataset, the researcher can save and load ODTK-Dataset as a binary file. Base on the experiment, the I/O speed for ODTK-Dataset binary file is 20 times faster than the raw data set file. Consider each study can have a different target to utilize, ODToolkit allows the researcher to add and use their evaluation metrics with the same rule as models. Additionally, to saving time for the researcher to adapt their model into ODToolkit, ODToolkit provides three folders for models, evaluation metrics, and data sets separately. The researcher can put the ODTK-Dataset binary file, formatted model, and evaluation metrics into the corresponding folder, then ODToolkit can automatically detect and adapt it into the toolkit.

Our work has several shortcomings that we intend to address in future work. Specifically, we have so far implemented a fraction of data-driven approaches proposed in related work to showcase how ODToolkit can be used to evaluate them. We plan to implement other data-driven approaches and extend the toolkit to grey-box and white-box models (*e.g.*, high-order physics-based occupancy detection models). We also intend to develop other domain-adaptive models and test them in a scenario where the source and target domains are located in two different buildings. Finally, we have not yet fully explored how flexible and user-friendly this toolkit is. We plan to conduct a survey among the potential users of this toolkit to solicit feedback, and understand their needs and expectations.

Chapter 5

Thermal Modelling

Thermal modelling is essential for intelligent control of building heating and cooling systems and for exploiting the thermal capacity of the building to offer a wide range of services to the electrical grid. In particular, thermal models explain how the indoor temperature responds to changes in the outside temperature and the operation of heating and cooling systems. Furthermore, it is possible to estimate *time-to-temperature*, i.e., when the indoor temperature reaches a desired temperature, or to estimate how much energy would be saved as a result of lowering the setpoint temperature or widening the *deadband*, i.e., the temperature range where the HVAC system does not operate.

There are several approaches to develop a thermal model which are discussed in Section 2.2. While an RC model is sufficiently accurate in most cases, the resistance and capacitance parameters are not typically known in practice. To identify these parameters, one must know the building's size, layout, structure and insulation. This information is not readily available for most residential buildings in operation today. Estimating the model parameters also requires a large amount of data. Another possibility is to take a data-driven approach to build a time-series model, e.g., SARIMAX, for predicting the temperature inside the building.

In recent years, smart thermostats, such as ecobee and Nest, have been installed in many homes and buildings. A smart thermostat measures the temperature inside and outside of a building at regular intervals. Additionally, users can enter the setpoint temperature and cooling/heating schedules via its

interface. Using the data collected by these thermostats, it is possible to build both physics-based and data-driven thermal models as we will explain in this chapter.

This chapter describes a methodology for building thermal models that addresses the above-mentioned challenges and presents our preliminary results for modelling temperature dynamics of residential buildings. We utilize Bayesian Neural Network to identify the RC model of a residential building, and compare the accuracy of this model with that of a SARIMAX model built using the same amount of data. Both models estimates the current indoor temperature by using current observations and historical data.

Our results indicate that the performance of the RC model is stable and does not depend on the granularity of data. Temperature predictions are generally accurate with a low RMSE of around 0.4 Fahrenheit. Meanwhile, the accuracy of a SARIMAX model relies heavily on data granularity. With fine-grained measurements, the accuracy is high. It reduces noticably when measurements are coarse-grained. If the smart thermostat collects data every 5 minutes, the RC model and SARIMAX model perform similarly.

The rest of this chapter is organized as follows. Section 5.1 describes our methodology that involves relaxing an RC model, estimating model parameters, and fitting a SARIMAX model to the time-series data. Section 5.2 explains the evaluation results, and Section 5.3 presents discussion and suggests directions for future work.

5.1 Methodology

A first-principle RC model requires the knowledge of building insulation and thermal mass which determine the values of resistance and capacitance parameters. However this information is not readily available for all homes. To address this problem, we adopt a grey-box modelling approach, that is we use the data acquired from smart thermostats to train Bayesian Neural Networks. These networks learn the physical relations governing the temperature of a building. In effect they eventually learn parameters of the RC model. We

also develop data-driven SARIMAX models to evaluate the so obtained RC models.

5.1.1 Grey-box modelling

Consider the 2nd-order RC model with 3 capacitors and 3 resistors which is illustrated in Figure 2.3. A set of differential equations (refer to Equation 2.1) explain the changes in the temperature inside the building based on the temperatures of other lumps and operation of the HVAC system. The state space representation of these differential equations is:

$$\frac{\partial \vec{x}}{\partial t} = \mathbf{A}\vec{x} + \mathbf{B}\vec{u} \tag{5.1}$$

$$\vec{y} = \mathbf{C}\vec{x} + \mathbf{D}\vec{u} \tag{5.2}$$

where $\vec{x} = \begin{bmatrix} T_1 & T_2 & T_{in} \end{bmatrix}^\top$ is a vector that represents the state, $\vec{y}$ is the measured temperature by a smart thermostat, $\vec{u} = \begin{bmatrix} T_{out} & k_{heat} & k_{cool} \end{bmatrix}^\top$ is a vector that represents the control input, and

$$\mathbf{A} = \begin{bmatrix} -\frac{1}{C_1R_1} - \frac{1}{C_1R_2} & \frac{1}{C_1R_2} & 0 \\ \frac{1}{C_2R_2} & -\frac{1}{C_2R_2} - \frac{1}{C_2R_3} & \frac{1}{C_2R_3} \\ 0 & \frac{1}{C_3R_3} & -\frac{1}{C_3R_3} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \frac{1}{C_1R_1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{Q_{heat}}{C_3} & -\frac{Q_{cool}}{C_3} \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^\top \qquad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}^\top$$

Here $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ are constant coefficient matrices. Equation 5.1 describes the state evolution over time given the input, and Equation 5.2 describes how the output is computed given the state and input. As shown in [98] the solution to this system of differential equations can be written as:

$$\vec{x_t} = (F \cdot \mathbf{I} - \Phi)^{-1}(F \cdot \Gamma_2 + \Gamma_1 - \Gamma_2)\vec{u}_t \tag{5.3}$$

where F denotes the forward shift operator (i.e., $F\vec{x_t} = \vec{x}_{t+\delta}$ where δ denotes the time step), $\Phi = e^{\mathbf{A}\delta}$ is the exponential matrix, and

$$\Gamma_1 = \mathbf{A}^{-1}(\Phi - \mathbf{I})\mathbf{B}$$
$$\Gamma_2 = \mathbf{A}^{-1}\left[\frac{\Gamma_1}{\delta} - \mathbf{B}\right]$$

Equation 5.3 relates the state at time $t + \delta$ to the state at t and the inputs at times t and $t + \delta$. Substituting $\vec{x_t}$ into Equation 5.2 yields:

$$\vec{y_t} = \mathbf{C}(F \cdot \mathbf{I} - \Phi)^{-1}(F \cdot \Gamma_2 + \Gamma_1 - \Gamma_2)\vec{u}_t + \mathbf{D}\vec{u}_t \tag{5.4}$$

Since $\mathbf{D} = \begin{bmatrix} 0 & 0 \end{bmatrix}$ we can rewrite the above equation as follows:

$$\vec{y_t} = \mathbf{C}(F \cdot \mathbf{I} - \Phi)^{-1}(F \cdot \Gamma_2 + \Gamma_1 - \Gamma_2)\vec{u}_t \tag{5.5}$$

To solve the equation, we first need to calculate the inverse of $(F \cdot \mathbf{I} - \Phi)$ matrix, which is equal to the adjoint of $(F \cdot \mathbf{I} - \Phi)$ divided by determinant of $(F \cdot \mathbf{I} - \Phi)$, assuming that its determinant is nonzero:

$$(F \cdot \mathbf{I} - \Phi)^{-1} = \frac{\mathbf{R}_0 F^2 + \mathbf{R}_1 F + \mathbf{R}_2}{F^3 + e_1 F^2 + e_2 F + e_3} \tag{5.6}$$

and then take the Equation 5.6 back to the Equation 5.5:

$$(F^3 + e_1 F^2 + e_2 F + e_3)\vec{y_t} = \mathbf{C}(\mathbf{R}_0 F^2 + \mathbf{R}_1 F + \mathbf{R}_2)(F \cdot \Gamma_2 + \Gamma_1 - \Gamma_2)\vec{u}_t$$

This can be written in a more compact form

$$\vec{y_t} = \sum_{i=0}^{3} \mathbf{S}_i \vec{u}_{t-i\delta} - \sum_{i=1}^{3} e_i \vec{y}_{t-i\delta} \tag{5.7}$$

where

$$\mathbf{S}_0 = \mathbf{C}\mathbf{R}_0\Gamma_2$$
$$\mathbf{S}_1 = \mathbf{C}[\mathbf{R}_0(\Gamma_1 - \Gamma_2) + \mathbf{R}_1\Gamma_2]$$
$$\mathbf{S}_2 = \mathbf{C}[\mathbf{R}_1(\Gamma_1 - \Gamma_2) + \mathbf{R}_2\Gamma_2]$$
$$\mathbf{S}_3 = \mathbf{C}\mathbf{R}_2(\Gamma_1 - \Gamma_2)$$

and

$$\mathbf{R}_0 = \mathbf{I} \qquad e_1 = -Tr(\Phi\mathbf{R}_0)$$
$$\mathbf{R}_1 = \Phi\mathbf{R}_0 + e_1\mathbf{I} \qquad e_2 = -Tr(\Phi\mathbf{R}_1)/2$$
$$\mathbf{R}_2 = \Phi\mathbf{R}_1 + e_2\mathbf{I} \qquad e_3 = -Tr(\Phi\mathbf{R}_2)/3$$

where $Tr()$ denotes the trace of a matrix.

Therefore, we get a linear relationship that maps the measured indoor temperatures T_{in} for the previous three time slots, and the outside temperatures and heat fluxes from the HVAC system for the current time slot and the previous three time slots to the indoor temperature of the current time slot (refer to Equation 5.7):

$$\vec{y}_t = f(\vec{u}_t, \vec{u}_{t-1}, \vec{u}_{t-2}, \vec{u}_{t-3}, \vec{y}_{t-1}, \vec{y}_{t-2}, \vec{y}_{t-3})$$

Here f is a linear function which can be learned using a Bayesian neural network. We refer to this neural network model as BNN-RC because it mimics the structure of the RC model presented in Section 2.2.1.

Variational Bayesian Learning

Consider the problem of finding a posterior distribution over some parameters, e.g., weights of a neural network denoted by a vector w. From the Bayes' theorem we have:

$$\begin{aligned} p(w|\mathcal{D}) &= \frac{p(\mathcal{D}, w)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})} \\ &= \frac{p(\mathcal{D}|w)p(w)}{\int p(\mathcal{D}|w)p(w)dw} \end{aligned} \tag{5.8}$$

where $\mathcal{D}$ is the training data set of size m

$$\mathcal{D} = (x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m),$$

$p(w|\mathcal{D})$ is the posterior probability, $p(\mathcal{D}, w)$ is the joint probability, $p(\mathcal{D}|w)$ is the likelihood, $p(w)$ is the prior probability, and $p(\mathcal{D})$ is the evidence. We note that computing the integral which appears in the denominator of Equation 5.8 is intractable for nontrivial problems.

Since calculating $p(w|\mathcal{D})$ is intractable, we can approximate it with the variational posterior $q(w|\theta)$ which is restricted to belong to a family of distributions simpler than $p(w|\mathcal{D})$. Variational learning finds the parameters θ of the variational posterior that minimize the dissimilarity between $p(w|\mathcal{D})$ and $q(w|\theta)$. We use Kullback-Leibler (KL) divergence of the true posterior from the variational posterior as the dissimilarity function:

$$\begin{aligned}\widehat{\theta} &= \arg\min_{\theta} \mathrm{KL}[q(w|\theta)||p(w|\mathcal{D})] \\ &= \arg\min_{\theta} \mathrm{KL}[q(w|\theta)||p(w)] - \mathbb{E}_{q(w|\theta)}[\log p(\mathcal{D}|w)]\end{aligned}$$

Hence, the dissimilarity function that is minimized is [19]:

$$\mathcal{F}(\mathcal{D},\theta) = \mathrm{KL}[q(w|\theta)||p(w)] - \mathbb{E}_{q(w|\theta)}[\log p(\mathcal{D}|w)]$$

Using Monte Carlo (MC) sampling, we can evaluate the expectation in this dissimilarity function and rewrite it as follows:

$$\mathcal{F}(\mathcal{D},\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \log q(w^{(i)}|\theta) - \log p(w^{(i)}) - \log p(\mathcal{D}|w^{(i)})$$

where $w^{(i)}$ is the i-th MC sample from the variational posterior, and N is the number of samples.

We can use a Gaussian prior for the parameters (e.g., weights of the neural network):

$$p(w) = \prod_i \mathcal{N}(w_i|0,\sigma_p^2)$$

,

$$\log p(w) = \sum_i \log \mathcal{N}(w_i|0,\sigma_p^2)$$

or a Gaussian scale mixture prior:

$$\begin{aligned} p(w) &= \prod_i (\pi\mathcal{N}(w_i|0,\sigma_1^2) + (1-\pi)\mathcal{N}(w_i|0,\sigma_2^2)) \\ \log p(w) &= \sum_i \log(\pi\mathcal{N}(w_i|0,\sigma_1^2) + (1-\pi)\mathcal{N}(w_i|0,\sigma_2^2)) \end{aligned} \tag{5.9}$$

The variational posterior when choosing the Gaussian distribution becomes:

$$q(w|\theta) = \prod_i \mathcal{N}(w_i|\mu,\sigma^2)$$

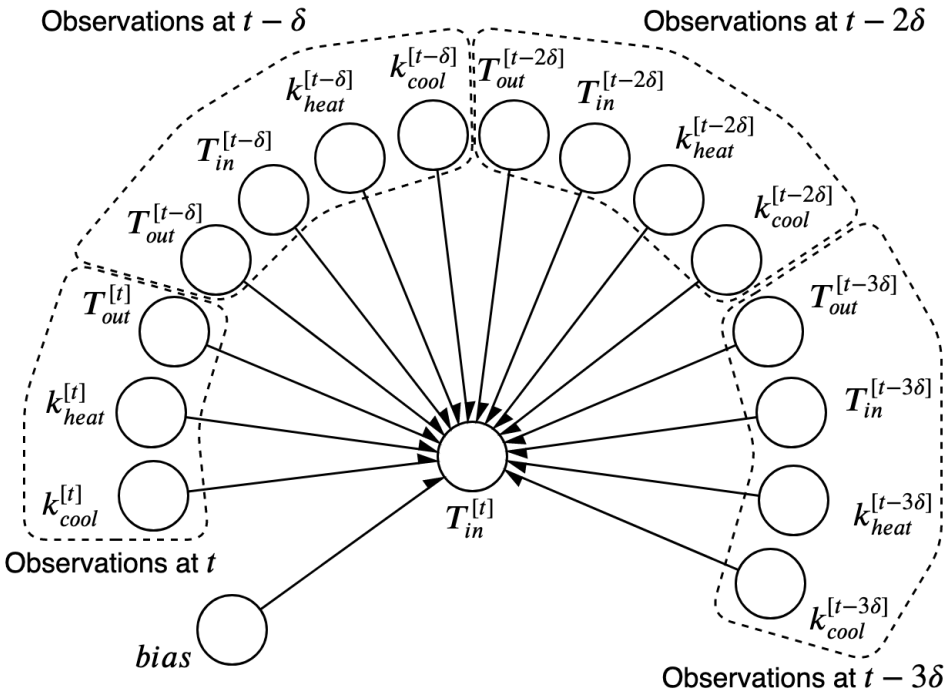


Figure 5.1: Structure of the BNN-RC model used to predict the indoor temperature.

$$\log q(w|\theta) = \sum_i \log \mathcal{N}(w_i|\mu, \sigma^2)$$

Finally, stochastic gradient descent is applied to optimize the variational parameters.

Bayesian Neural Networks

Artificial neural networks have been quite successful in numerous learning tasks. But sometimes they suffer from over-fitting and cannot effectively classify data, especially if this data has not been seen before. To address these limitations, a Bayesian neural network (BNN) introduces uncertainty in the weights and biases of the network [19]. The benefit of adding uncertainty is two-fold. First, this uncertainty furnishes the neural network with the ability to recognize any unseen pattern, and give more reasonable predictions when similar data has not been seen before. Second, this uncertainty counters the issue of over-fitting, providing a means for selecting optimal neural network model [55], [68]. Besides, as discussed by [86], the Bayesian approach offers

other advantages when it comes to building an RC model.

In BNN, weights are represented by probability distributions rather than having a single fixed value. The probability distributions are learned in a way that explains variability in the training data. The proposed method trains an ensemble of networks, where each network has its weights drawn from the learned probability distributions.

Drawing on variational Bayesian learning, it is possible to define a probability distribution over the weights of an artificial neural network and estimate the posterior probability given the prior probability and the likelihood.

We specifically adopt the Bayes-by-Backprop algorithm [19] which uses Monte Carlo sampling to obtain unbiased estimates of gradients of the dissimilarity function to learn a distribution over the weights of the neural network. The proposed method trains an ensemble of networks, where each network has its weights drawn from the learned probability distribution.

We develop a BNN with the model structure suggested by the RC model. We refer to this model as BNN-RC. To estimate the current value of indoor temperature, we use three previous outside temperature values and the measured indoor temperature, and also the HVAC control state k_{heat} and k_{cool}, plus the current value of the HVAC control state and outside temperature as input to the model. Thus, as Figure 5.1 shows, the BNN-RC model consists of 15 input nodes and one bias node, no hidden layer, and one output node. This makes the model linear, matching exactly the structure suggested by the RC model. For each neuron, we initialize its weight using a Gaussian scale mixture prior presented in Equation 5.9) with $\sigma_1 = 1$, $\sigma_2 = 0.1$, and $\pi = 0.2$. We note that the BNN-RC model does not recover the R-C parameters that pertain to this home; it simply maps the features (15 input variables) to the indoor temperature with a linear relationship that resembles the RC model (see Equation 5.7). The average weight of each neuron yields the coefficient of the corresponding input variable in the RC model.

5.1.2 Estimating parameters of SARIMAX

The SARIMAX model is previously explain in Section 2.2. In this section, we aim to address how the parameters of this model can be estimated from data. Temperature time-series data from a randomly selected home is utilized to showcase the process.

Estimating d for integrated part of the model requires analyzing the time-series data and determining whether the data is stationary or not. We have two types of stationarity, namely strict-sense and wide-sense. Let $\mathbf{F_X}$ represent the joint distribution and X_t represent a stochastic time-series, if the following equation holds, the time-series is strict-sense stationary:

$$\mathbf{F_X}(x_{t_1+\tau}, x_{t_2+\tau}, \cdots, x_{t_n+\tau}) = \mathbf{F_X}(x_{t_1}, x_{t_2}, \cdots, x_{t_n})$$

$$\text{for all } \tau, t_1, \cdots t_n \in \mathbb{R}$$

Similarly, wide-sense stationarity is defined as follows:

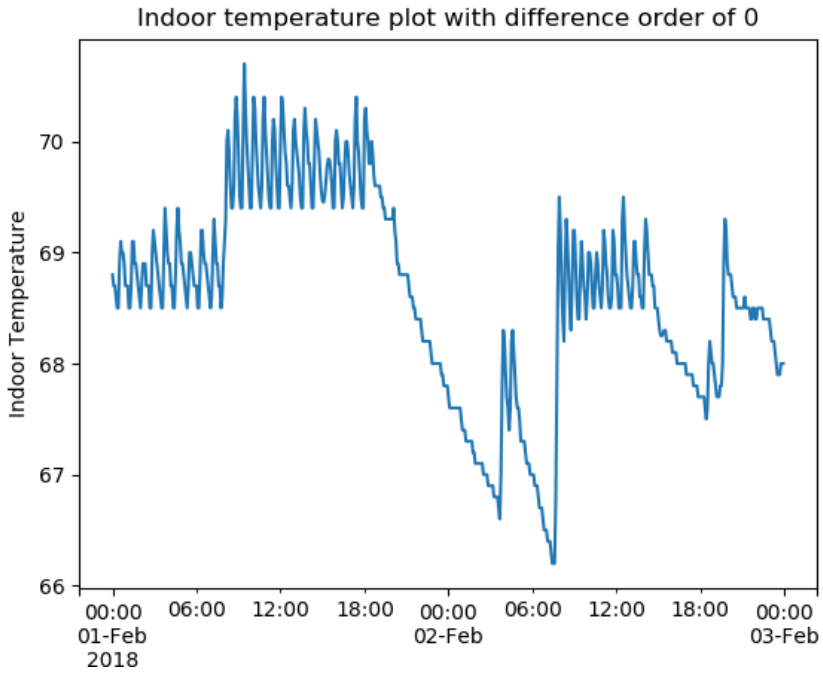
$$\begin{aligned} m_X(t) &= m_X(t+\tau) && \text{for all } \tau \in \mathbb{R} \\ K_{XX}(t_1, t_2) &= K_{XX}(t_1 - t_2, 0) && \text{for all } t_1, t_2 \in \mathbb{R} \\ \mathbb{E}[|X(t)|^2] &< \infty && \text{for all } t \in \mathbb{R} \end{aligned}$$

where

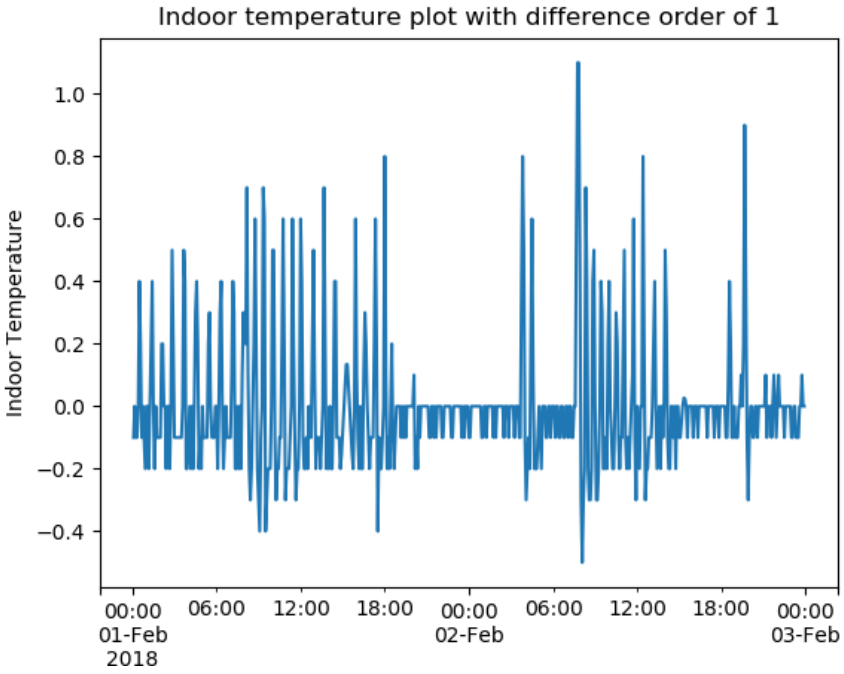
$$m_X(t) = \mathbb{E}[X_t]$$

$$K_{XX}(t_1, t_2) = \mathbb{E}[(X_{t_1} - m_X(t_1))(X_{t_2} - m_X(t_2))]$$

Thus, we iteratively apply the differencing operator discussed in Section 2.2 (increasing d from 0) and each time we check if the differenced time-series has become stationary. To check stationarity of the differenced time-series, we use the Augmented Dickey-Fuller (ADF) test - a formal statistical test for stationarity. This test helps to identify whether the time-series has a unit root (i.e., a stochastic trend in the time-series). The null hypothesis is that a unit root is present in the time-series, which is rejected when the p-value of ADF is lower than or equal to 0.05; this indicates that the time-series is stationary. Moreover, the ADF statistic is a negative number that represents



(a)



(b)

Figure 5.2: Sample time-series data before and after performing differencing.

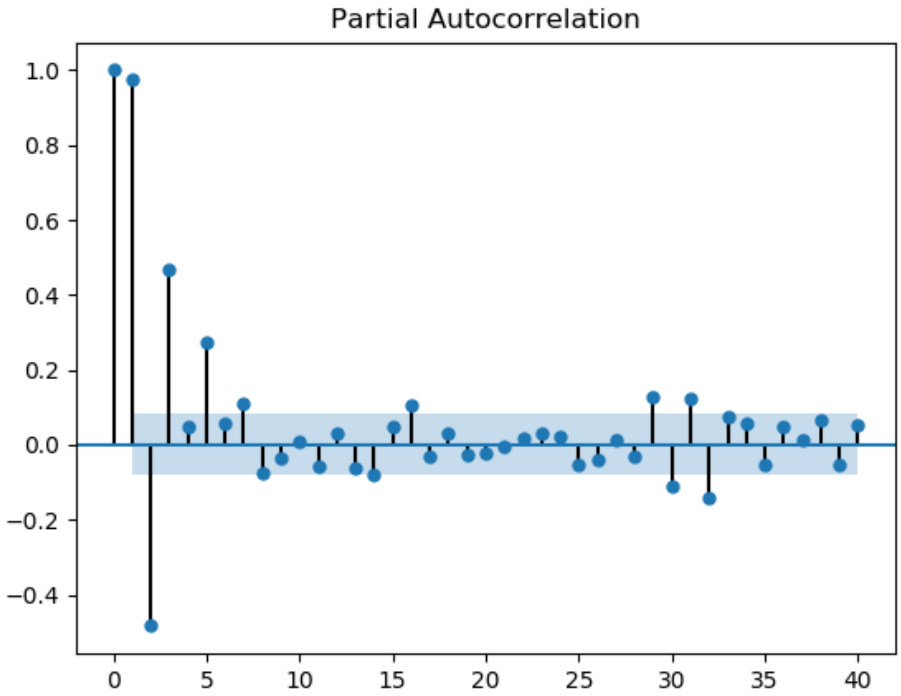


Figure 5.3: PACF calculated for a sample temperature time-series.

the evidence for rejecting the null hypothesis at some confidence level. The more negative this number is, the stronger is the evidence for rejecting the null hypothesis. Figure 5.2 shows the original time-series and the differenced time-series ($d = 1$), and the ADF test suggest the parameter d as 1.

To estimate p of the autoregressive part, it is essential to draw the partial autocorrelation function (PACF). PACF shows the lagged correlation between time-series values, after excluding intermediate lags. It is defined as follows:

$$\text{PACF}(X, k) = \begin{cases} corr(X_2, X_1) & \text{, for } k = 1 \\ corr(X_{t+k+1} - P_{t,k}(X_{t+k+1}), X_{t+1} - P_{t,k}(X_{t+1})) & \text{, for } k \geq 2 \end{cases}$$

where k is the lag, which is 5 minutes in this sample data set, and $P(X)$ is the surjective operator of the orthogonal projection of X onto the linear space of Hilbert space spanned by $X_{t+1}, \cdots, X_{t+k}$.

Looking at the PACF, which is shown Figure 5.3, it can be concluded that p should be set to 4 as the first value falls into the confidence interval.

To estimate q for the moving average part, it is crucial to draw the au-

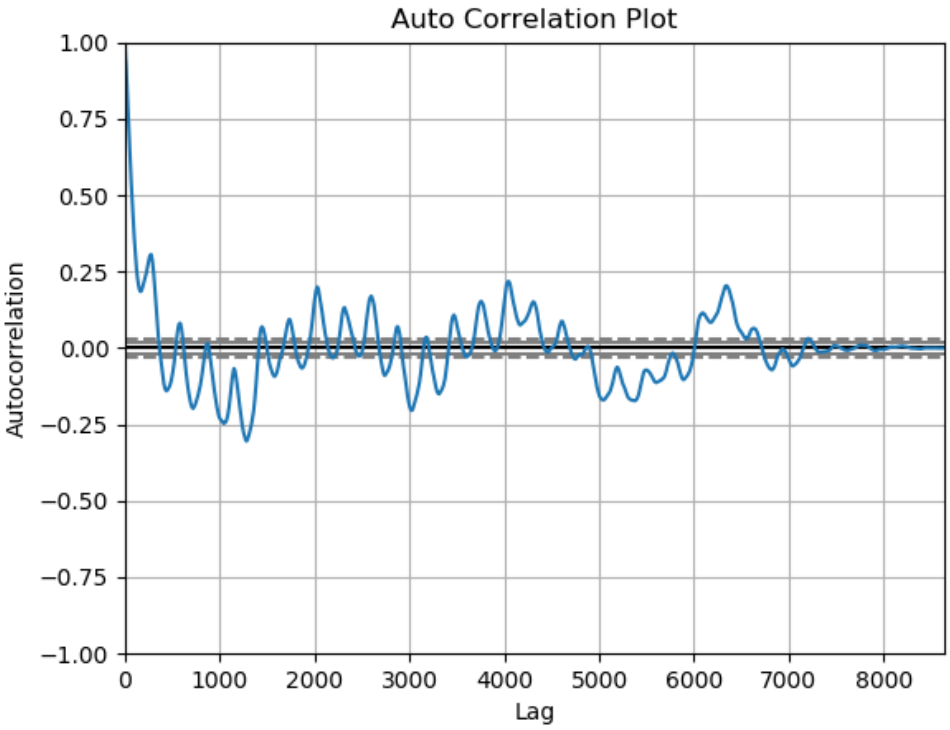


Figure 5.4: ACF calculated for a sample temperature time-series.

tocorrelation function (ACF). ACF suggests how many terms the MA model needs to remove autocorrelation in the time-series. ACF is defined as follow:

$$\mathrm{ACF}(k) = \frac{\frac{1}{n-k}\sum_{t=k+1}^{n}(T_t - \bar{T})(T_{t-k} - \bar{T})}{\sqrt{\frac{1}{n}\sum_{t=1}^{n}(T_t - \bar{T})}\sqrt{\frac{1}{n-k}\sum_{t=k+1}^{n}(T_{t-k} - \bar{T})}}$$

Looking at the ACF, which is shown in Figure 5.4, we set q to 12.

We adopt the same features used in the grey-box RC model as explanatory variables in the SARIMAX model. These variables are the HVAC operation mode and outside temperature. Turning out attention to the seasonal effect, Figure 5.4 suggests that diurnal and weekly effects exist in the temperature time-series. Thus, we have to remove these effects by decomposing those effects. We apply a convolution filter to the data to identify the seasonal component [41]. Figure 5.5 shows the resulting time-series after removing the diurnal effect and Figure 5.6 shows the resulting time-series after removing the weekly effect. We repeat the same process for all homes in order to build well-suited SARIMAX models.

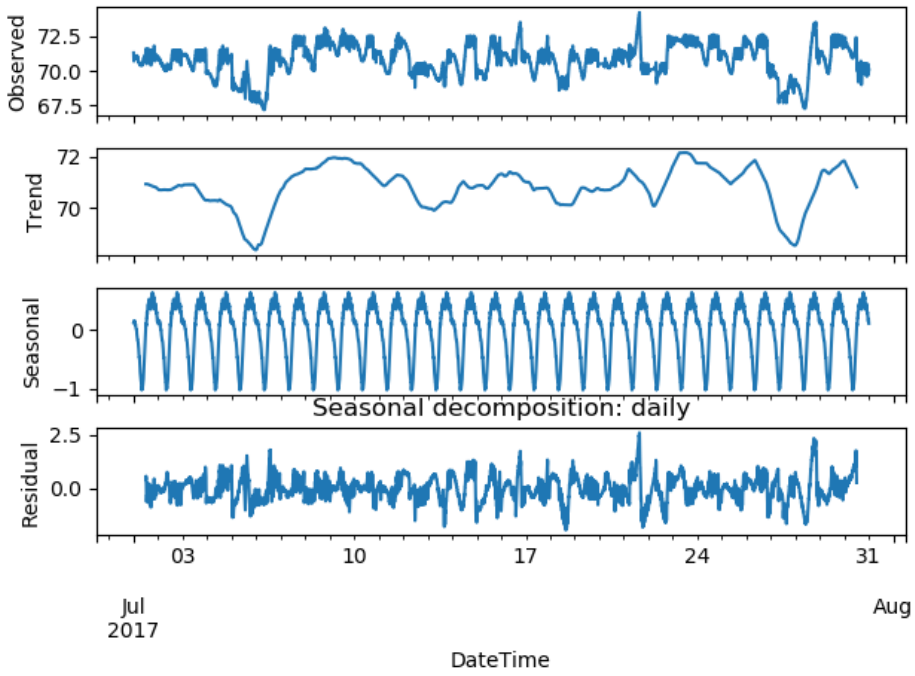


Figure 5.5: Resulting time-series after decomposing and removing the daily effect.

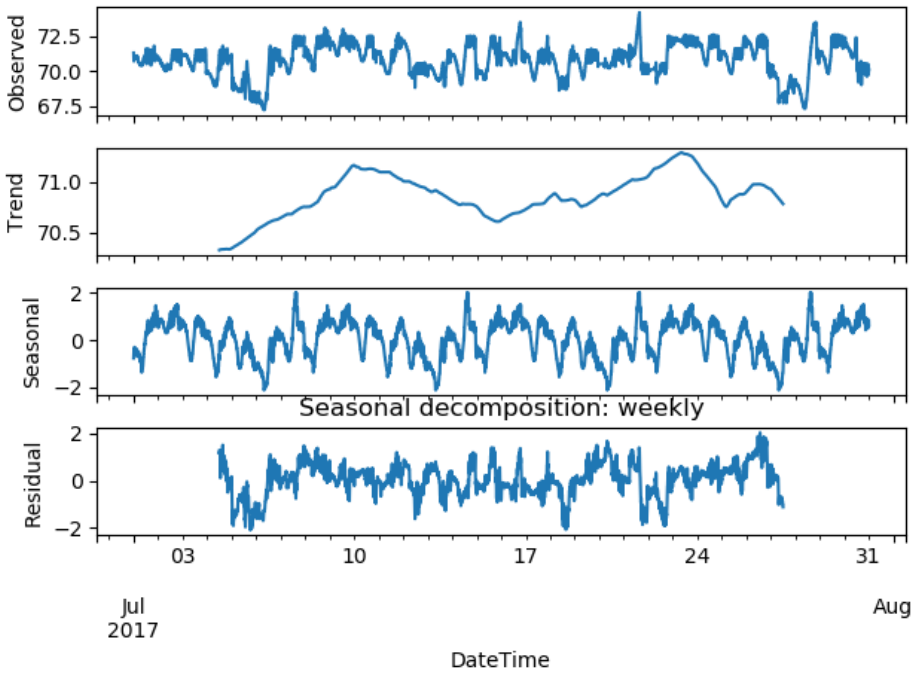


Figure 5.6: Resulting time-series after decomposing and removing the weekly effect.

Table 5.1: Performance of the SARIMAX model in two randomly selected locations

	Location A	**Location B**
MSE	0.0695	0.0615
RMSE	0.263	0.241
MAPE	0.1765%	0.176%

5.2 Results

In this section, we discuss the accuracy of the proposed grey-box RC model and SARIMAX model, assuming that both of them are built for each home as explained earlier. The evaluation metrics are defined in Section 3.2.

5.2.1 Data sets

We obtained data from over 7000 homes in Canada from ecobee smart thermostats. The data set contained homes with 3 months to 3 years worth of data. For each home we had the city it is located in, the indoor temperature, the setpoint of the HVAC system, and the outdoor temperature, all of them recorded at 5-minute intervals.

5.2.2 Evaluating performance of the SARIMAX model

Table 5.1 shows the result of predicting temperature using the SARIMAX models built for two different geographical locations. For each location we randomly selected five homes and computed the mean of their prediction accuracy. All selected homes contain at least two years of data. We used 3 months of data to train the model, and then used it to predict the temperature for the rest of the time.

From Table 5.1, the model predicts the temperature relatively accurately, with RMSE around 0.25 and MAPE of 0.17%. This result motivates the use of black-box models. Also, increasing the amount of available training data, does not affect the accuracy of this model in a consistent manner. We attribute this to the fact that the SARIMAX model learned to predict the temperature using

a value that is very similar to the previous value, irrespective of the amount of training data that was available. Since the sensor reported data every 5 minutes, the change of temperature between two consecutive intervals is very small, explaining the good performance of the SARIMAX model.

Furthermore, we train the SARIMAX model on all homes available using 3 months of data to train the model, and then test on all available data we have. The average RMSE across all homes after filtering out significant outliers (by using the rule of 1.5 interquartile range) is 0.3454. Figure 5.7 shows the distribution plot of the RMSE.

5.2.3 Evaluating performance of the grey-box RC model

We evaluate the grey-box RC model by considering its performance on the training and test data sets both pertaining to the same home. Because BNN uses Gaussian distributions to represent weights, the estimation accuracy is different for each run even for the same model. To get an accurate and reliable result, we test the model with 5 runs and 50 runs and compute the average result in each case. The differences between the average RMSE of 5 runs and the average RMSE of 50 runs are only 0.037 on all cases with different settings, which is equivalent to an nRMSE of only 0.000046. But the runtime for 5 runs only takes 1 minute for each home, while carrying out 50 runs requires almost 7 minutes for each home on Core i7-7700 CPU. Therefore, we use the average RMSE of 5 runs in the rest of this thesis.

First, we evaluate the average performance of building models on a given home with enough available training data. The average RMSE of the model is 0.3471 when using three-month data for training, which is slightly worse than the SARIMAX model. Figure 5.7 shows how the RMSE is distributed for BNN mdoel. Second, we explore how the RMSE changes as the amount of training data changes. The result shows that when only one-day data is available, the average RMSE is 44.584, and when two months of training data is available, the average RMSE is 0.35. Therefore the RMSE always decreases when training data is abundant.

5.2.4 Comparing black-box and grey-box models

Figure 5.7 demonstrates the distribution of RMSE values for the two proposed thermal models, SARIMAX and BNN-RC, when considering all homes in Canada. The SARIMAX model has a narrower distribution of RMSE values compared to the BNN-RC model. Furthermore, the distribution of RMSE values of the BNN-RC model is skewed to the left compared to that of the SARIMAX model, which suggests that BNN-RC significantly outperforms SARIMAX for many homes.

Figure 5.8 shows a real estimation result on a non-exist average home[1]. The naive baseline in this experiment estimates the same value for the current indoor temperature as the the previous indoor temperature. Both RC and SARIMAX models are trained with 3 months worth of data. The figure shows that although SARIMAX has a lower RMSE compared to grey-box RC model, the estimation is usually delayed, whereas the grey-box RC model estimates the trend correctly. Therefore, we can conclude that grey-box RC model performs better in general.

5.3 Discussion

Despite the extensive body of research on thermal modelling, few related works consider the temperature as the model output. Most studies focus on the heat flux. This chapter demonstrated why the data-driven model works surprisingly well by extending and deriving the open form of the RC model. Moreover, we leveraged a BNN to accomplish the thermal modelling task for the first time and obtained very low average RMSE of 0.3471. We finally concluded that grey-box models slightly outperform black-box models, highlighting the fact that knowing the physical relations can help to increase the modelling accuracy.

We only used data from homes located in Canada in this preliminary work. One possible direction for future work is to generate and evaluate both types of models for other countries to see whether there is any meaningful difference when considering a different climate. Moreover, further research should be un-

[1]We cannot share results for a real home due to privacy reasons.

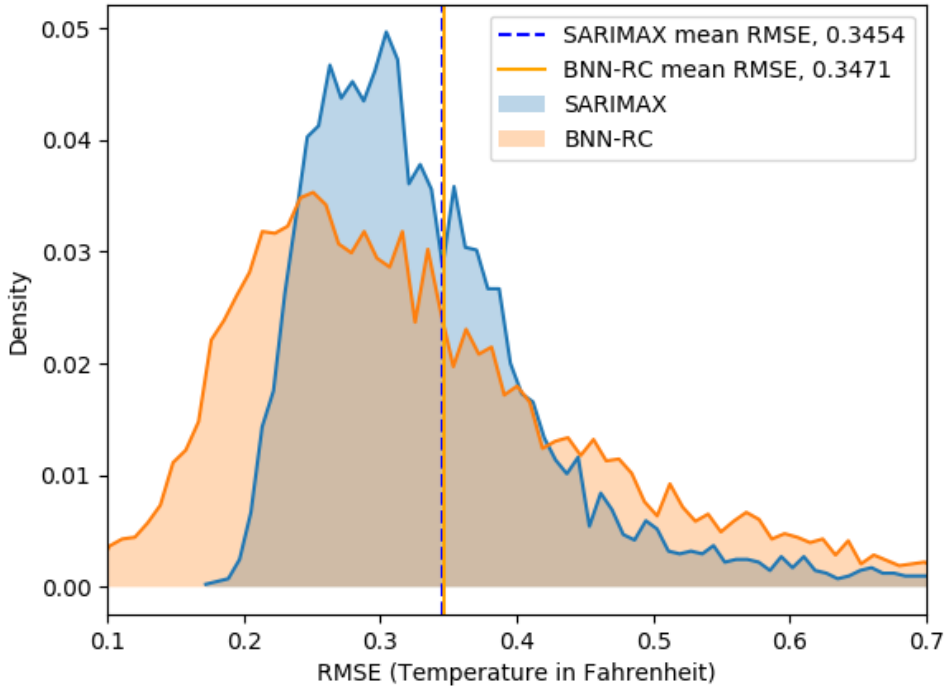


Figure 5.7: Distribution of RMSE values for all homes in our data set.

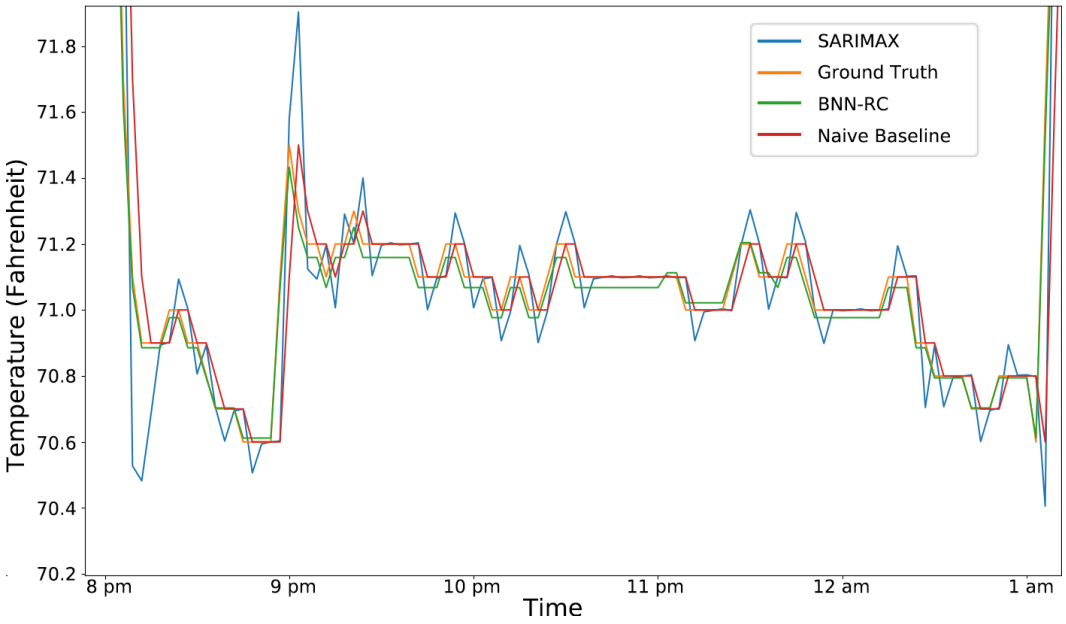


Figure 5.8: Estimation result comparison.

dertaken to investigate the possibility of retrieving the actual RC values from the BNN weights. The combined RC values can only work for this temperature estimation model. Determine the actual RC values can be useful in other applications, such as the ones that require analyzing the building envelope. Last but not least, we aim to apply transfer learning to facilitate model development for homes that are not equipped with a smart thermostat. BNN is an appropriate framework for transfer learning because the posterior learned in the source domain can be used as the prior in the target domain. If the two domains are similar, utilizing the posterior probability from a well-trained model should be better than simply adopting a Gaussian distribution as the prior probability.

Chapter 6

Conclusion

Intelligent control of building subsystems could provide a more comfortable environment for the occupants while reducing the energy consumption of the whole building. Most advanced control techniques are proactive in the sense that they predict occupancy and temperature dynamics of the built environment, and utilize the thermal capacity of the building to *shape* the demand of the HVAC system. Thus, the occupancy and thermal models play a key role in these *model-based* control techniques. This thesis studied comprehensive occupancy and thermal modelling for residential and commercial buildings to support advanced control applications and investigated the feasibility of applying transfer learning to enable, facilitate, and speed up model development, especially when training data is sparse or non-existent.

The contribution of this thesis is as follows:

- We developed accurate data-driven models for occupancy estimation when sufficient ground truth data is available. Training these models requires only HVAC sensor data and weather data, and the obtained models are capable of describing the complex, nonlinear relationship between historical sensor data and the number of occupants, and the temporal dependence of the occupancy data.

- We adopted semi-supervised and unsupervised domain-adaptation techniques to reduce the amount of ground truth data required for developing a well-suited model.

- We evaluated the efficacy of these models in different settings and studied how the prediction accuracy would change with the amount of available ground truth data.

- We presented the design and implementation of ODToolkit and discussed how it can be extended to incorporate new data sets, algorithms, and metrics. We extended the toolkit with three new semi-supervised domain-adaptive occupancy detection algorithms and evaluated their performance by comparing them against the state-of-the-art supervised learning algorithms which are also implemented in the toolkit. This way, we justified the effort to obtain reliable occupancy labels even for a short period of time.

- We investigated how using the toolkit reduces the time and effort required to build new models and addressed important research questions, such as what algorithms perform best in various occupancy detection tasks, and what sensing modalities and how much ground truth data are necessary to train a model that could achieve a sufficiently high level of accuracy.

- We used Bayesian neural networks to identify the parameters of a 2nd RC model without assuming the knowledge about the building insulation and thermal mass.

- We compared the prediction accuracy of the grey-box RC model with the SARIMAX model when both are trained using the limited training data that was available.

For future work, we plan to focus on evaluating the performance of these models when it comes to predicting the room temperature and its occupancy state several hours in advance. Moreover, we intend to investigate how transfer learning can be applied to building thermal models. Additionally, we aim to study how the developed models can be incorporated in learning-based control of building subsystems and compare the performance of model-based

and model-free control techniques for HVAC and lighting systems. We also plan to deploy these control applications in real buildings and empirically validate our results.

References

[1] Y. Agarwal *et al.*, "Duty-cycling buildings aggressively: The next frontier in HVAC control," in *Proc. 10th International Conference on Information Processing in Sensor Networks (IPSN)*, ACM/IEEE, Apr. 2011, pp. 246–257. 2, 10, 12

[2] B. Ai, Z. Fan, and R. X. Gao, "Occupancy estimation for smart buildings by an auto-regressive hidden markov model," in *American Control Conference (ACC), 2014*, IEEE, 2014, pp. 2234–2239. 14

[3] I. B. A. Ang, F. D. Salim, and M. Hamilton, "Human occupancy recognition with multivariate ambient sensors," in *International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, IEEE, 2016, pp. 1–6. 10

[4] O. Ardakanian, A. Bhattacharya, and D. Culler, "Non-intrusive techniques for establishing occupancy related energy savings in commercial buildings," in *Proc. 3rd International Conference on Systems for Energy-Efficient Built Environments*, ACM, 2016, pp. 21–30. 76

[5] ——, "Non-intrusive occupancy monitoring for energy conservation in commercial buildings," *Energy and Buildings*, vol. 179, pp. 311–323, 2018. 2, 3, 11, 12

[6] K. Arendt *et al.*, "Room-level occupant counts, airflow and co2 data from an office building," in *Proc. 1st Workshop on Data Acquisition To Analysis*, 2018, pp. 13–14. 36, 59, 60

[7] I. B. Arief-Ang, M. Hamilton, and F. D. Salim, "A scalable room occupancy prediction with transferable time series decomposition of co2 sensor data," *ACM Trans. Sen. Netw.*, vol. 14, no. 3-4, 21:1–21:28, Nov. 2018, ISSN: 1550-4859. 10, 12

[8] ——, "RUP: Large room utilisation prediction with carbon dioxide sensor," *Pervasive and Mobile Computing*, vol. 46, pp. 49–72, 2018. 10, 12

[9] I. B. Arief-Ang, F. D. Salim, and M. Hamilton, "Da-hoc: Semi-supervised domain adaptation for room occupancy prediction using co2 sensor data," in *Proc. 4th International Conference on Systems for Energy-Efficient Built Environments (BuildSys)*, ACM, 2017, 1:1–1:10. 3, 30, 31

[10] A. Aswani *et al.*, "Reducing transient and steady state electricity consumption in hvac using learning-based model-predictive control," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 240–253, 2012. 3, 11

[11] C. Baglivo, P. Congedo, M. Di Cataldo, L. Coluccia, and D. D'Agostino, "Envelope design optimization by thermal modelling of a building in a warm climate," *Energies*, vol. 10, no. 11, p. 1808, 2017. 22, 23

[12] B. Balaji *et al.*, "Sentinel: Occupancy based HVAC actuation using existing WiFi infrastructure within commercial buildings," in *Proc. 11th Conference on Embedded Networked Sensor Systems (SenSys)*, ACM, 2013, 17:1–17:14. 11, 12

[13] ——, "Brick : Metadata schema for portable smart building applications," *Applied Energy*, vol. 226, pp. 1273–1292, 2018. 65

[14] S. Barker *et al.*, "Smart*: An open data set and tools for enabling research in sustainable homes," *SustKDD, August*, vol. 111, no. 112, p. 108, 2012. 59, 60

[15] C. Basu, C. Koehler, K. Das, and A. K. Dey, "Perccs: Person-count from carbon dioxide using sparse non-negative matrix factorization," in *Proc. International Joint Conference on Pervasive and Ubiquitous Computing*, ACM, 2015, pp. 987–998. 10, 12, 16

[16] N. Batra *et al.*, "Nilmtk: An open source toolkit for non-intrusive load monitoring," in *Proc. 5th international conference on Future energy systems*, ACM, 2014, pp. 265–276. 5, 32, 56

[17] A. Beltran, V. L. Erickson, and A. E. Cerpa, "ThermoSense: Occupancy thermal based sensing for HVAC control," in *Proc. 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings (BuildSys)*, ACM, 2013, 11:1–11:8. 2, 10, 12

[18] T. Berthou *et al.*, "Smart-e: A tool for energy demand simulation and optimization at the city scale," in *Proceedings of the 14th Conference of International Building Performance Simulation Association*, 2015. 22, 23

[19] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, JMLR.org, 2015, pp. 1613–1622. 89–91

[20] J. E. Braun and N. Chaturvedi, "An inverse gray-box model for transient building load prediction," *HVAC&R Research*, vol. 8, no. 1, pp. 73–99, 2002. 23, 26

[21] J. S. Bridle and S. J. Cox, "Recnorm: Simultaneous normalisation and classification applied to speech recognition," in *Advances in Neural Information Processing Systems*, 1991, pp. 234–240. 30

[22] E. Cadenas, W. Rivera, R. Campos-Amezcua, and C. Heard, "Wind speed prediction using a univariate arima model and a multivariate narx model," *Energies*, vol. 9, no. 2, p. 109, 2016. 20

[23] L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models," *Energy and Buildings*, vol. 112, pp. 28–39, 2016. 12, 15, 57, 59, 60, 65, 72

[24] L. M. Candanedo, V. Feldheim, and D. Deramaix, "A methodology based on hidden markov models for occupancy detection and a case study in a low energy residential building," *Energy and Buildings*, vol. 148, pp. 327–341, 2017. 16

[25] W.-K. Chang and T. Hong, "Statistical analysis and modeling of occupancy patterns in open-plan offices using measured lighting-switch data," *Building Simulation*, vol. 6, no. 1, pp. 23–32, 2013. 2

[26] C. Chatfield, *The analysis of time series: an introduction*. Chapman and Hall/CRC, 2003. 26

[27] D. Chen *et al.*, "Non-intrusive occupancy monitoring using smart meters," in *Proc. 5th Workshop on Embedded Systems For Energy-Efficient Buildings*, ACM, 2013, pp. 1–8. 12, 59

[28] K. Cho *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014. 17

[29] K. Christensen *et al.*, "Using existing network infrastructure to estimate building occupancy and control plugged-in devices in user workspaces," *International Journal of Communication Networks and Distributed Systems*, vol. 12, no. 1, pp. 4–29, 2014. 11

[30] A. Conneau *et al.*, "Word translation without parallel data," *arXiv preprint arXiv:1710.04087*, 2017. 30

[31] A.-H. Deconinck and S. Roels, "Comparison of characterisation methods determining the thermal resistance of building components from onsite measurements," *Energy and Buildings*, vol. 130, pp. 309–320, 2016. 26

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255. 29

[33] S. Depatla, A. Muralidharan, and Y. Mostofi, "Occupancy estimation using only wifi power measurements," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 7, pp. 1381–1393, 2015. 11, 12

[34] B. Du, G. Verbic, and J. Fletcher, "Thermal modelling for demand response of residential buildings," in *2017 Australasian Universities Power Engineering Conference (AUPEC)*, IEEE, 2017, pp. 1–6. 22, 23

[35] A. Ebadat *et al.*, "Estimation of building occupancy levels through environmental signals deconvolution," in *Proc. 5th Workshop on Embedded Systems For Energy-Efficient Buildings*, ACM, 2013, pp. 1–8. 15, 16

[36] U. EIA, "Residential energy consumption survey, 2015 recs survey data," *Tables HC6*, vol. 8, 2015. 2

[37] V. L. Erickson, S. Achleitner, and A. E. Cerpa, "POEM: Power-efficient occupancy-based energy management system," in *Proc. 12th International Conference on Information Processing in Sensor Networks*, ACM, 2013, pp. 203–216. 10, 12

[38] M. Fayazbakhsh, F. Bagheri, and M. Bahrami, "A resistance–capacitance model for real-time calculation of cooling load in hvac-r systems," *Journal of Thermal Science and Engineering Applications*, vol. 7, no. 4, p. 041 008, 2015. 23, 26

[39] F. Fiebig, S. Kochanneck, I. Mauser, and H. Schmeck, "Detecting occupancy in smart buildings by data fusion from low-cost sensors: Poster description," in *Proc. 8th International Conference on Future Energy Systems*, ACM, 2017, pp. 259–261. 59, 60

[40] M. File, "Commercial buildings energy consumption survey (cbecs)," 2015. 2

[41] B. Fischer, *Decomposition of time series: comparing different methods in theory and practice*. Eurostat, 1995. 95

[42] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," *arXiv preprint arXiv:1409.7495*, 2014. 31, 32

[43] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016. 31

[44] T. Gebru, J. Hoffman, and L. Fei-Fei, "Fine-grained recognition in the wild: A multi-task domain adaptation approach," in *Proc. International Conference on Computer Vision (ICCV)*, IEEE, 2017, pp. 1358–1367. 5, 30

[45] S. K. Ghai, L. V. Thanayankizil, D. P. Seetharam, and D. Chakraborty, "Occupancy detection in commercial buildings using opportunistic context sources," in *Proc. International Conference on Pervasive Computing and Communications*, IEEE, Mar. 2012, pp. 463–466. 11, 12

[46] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, e215–e220, 2000. 33

[47] S. Golestan, S. Kazemian, and O. Ardakanian, "Data-driven models for building occupancy estimation," in *Proceedings of the Ninth International Conference on Future Energy Systems*, ACM, 2018, pp. 277–281. 11, 12, 16, 59

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016. 32

[49] Google, *Cloud AutoML*, Online `https://cloud.google.com/automl/`, 2018. (visited on 2018). 56

[50] M. Gouda, S. Danaher, and C. Underwood, "Building thermal model reduction using nonlinear constrained optimization," *Building and Environment*, vol. 37, no. 12, pp. 1255–1265, 2002. 24

[51] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2013, pp. 6645–6649. 17

[52] H. Hagras, V. Callaghan, M. Colley, and G. Clarke, "A hierarchical fuzzy–genetic multi-agent architecture for intelligent buildings online learning, adaptation and control," *Information Sciences*, vol. 150, no. 1-2, pp. 33–57, 2003. 54

[53] F. Haldi and D. Robinson, "The impact of occupants' behaviour on building energy demand," *Journal of Building Performance Simulation*, vol. 4, no. 4, pp. 323–338, 2011. 22, 23

[54] V. Harish and A. Kumar, "Reduced order modeling and parameter identification of a building energy system model through an optimization routine," *Applied Energy*, vol. 162, pp. 1010–1023, 2016. 23, 25

[55] H. S. Hippert and J. W. Taylor, "An evaluation of bayesian techniques for controlling model complexity and selecting inputs in a neural network for short-term load forecasting," *Neural networks*, vol. 23, no. 3, pp. 386–395, 2010. 90

[56] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 18

[57] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018. 5

[58] M. Jin *et al.*, "Occupancy detection via environmental sensing," *Transactions on Automation Science and Engineering*, 2016. 3

[59] M. Jin, N. Bekiaris-Liberis, K. Weekly, C. J. Spanos, and A. M. Bayen, "Occupancy detection via environmental sensing," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 443–455, 2018. 11, 12

[60] D. Jung *et al.*, "EnergyTrack: Sensor-driven energy use analysis system," in *Proc. 5th Workshop on Embedded Systems For Energy-Efficient Buildings (BuildSys)*, ACM, 2013, 6:1–6:8. 10, 12

[61] S. Kakaç, Y. Yener, and C. P. Naveira-Cotta, *Heat conduction*. CRC Press, 2018. 4, 22

[62] T. Kim *et al.*, "Learning to discover cross-domain relations with generative adversarial networks," *arXiv preprint arXiv:1703.05192*, 2017. 30

[63] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 20

[64] M. B. Kjærgaard, M. Werner, F. C. Sangogboye, and K. Arendt, "Dcount-a probabilistic algorithm for accurately disaggregating building occupant counts into room counts," in *International Conference on Mobile Data Management*, 2018. 10, 12

[65] D. Kotz and T. Henderson, "Crawdad: A community resource for archiving wireless data at dartmouth," *Pervasive Computing*, vol. 4, no. 4, pp. 12–14, 2005. 33

[66] J. Langevin, P. L. Gurian, and J. Wen, "Tracking the human-building interaction: A longitudinal field study of occupant behavior in air-conditioned offices," *Journal of Environmental Psychology*, vol. 42, pp. 94–115, 2015. 59, 60

[67] G. Laput, Y. Zhang, and C. Harrison, "Synthetic sensors: Towards general-purpose sensing," in *Proc. Conference on Human Factors in Computing Systems (CHI)*, ACM, 2017, pp. 3986–3999. 10

[68] P. Lauret, E. Fock, R. N. Randrianarivony, and J.-F. Manicom-Ramsamy, "Bayesian neural network approach to short time load forecasting," *Energy conversion and management*, vol. 49, no. 5, pp. 1156–1166, 2008. 90

[69] M. Lauster, J. Teichmann, M. Fuchs, R. Streblow, and D. Mueller, "Low order thermal network models for dynamic simulations of buildings on city district scale," *Building and Environment*, vol. 73, pp. 223–231, 2014. 22, 23

[70] Y. Low *et al.*, "Graphlab: A new framework for parallel machine learning," *arXiv preprint arXiv:1408.2041*, 2014. 33, 56

[71] M. Maasoumy, M. Razmara, M. Shahbakhti, and A. S. Vincentelli, "Handling model uncertainty in model predictive control for energy efficient buildings," *Energy and Buildings*, vol. 77, pp. 377–392, 2014. 22, 23

[72] A. Mahdavi and M. Taheri, "An ontology for building monitoring," *Journal of Building Performance Simulation*, vol. 10, no. 5-6, pp. 499–508, 2017. 65

[73] S. Mamidi, Y.-H. Chang, and R. Maheswaran, "Improving building energy efficiency with a network of sensing, learning and prediction agents," in *Proc. 11th International Conference on Autonomous Agents and Multiagent Systems*, Valencia, Spain: IFAAMAS, 2012, pp. 45–52. 10, 12

[74] ——, "Improving building energy efficiency with a network of sensing, learning and prediction agents," in *Proc. 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 45–52. 15

[75] D. P. Mandic and J. Chambers, *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. John Wiley & Sons, Inc., 2001. 21

[76] T. Mikolov *et al.*, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010. 17

[77] M. Mitchell, "An introduction to genetic algorithm.-mit press, 1996," 1996. 23

[78] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on neural networks*, vol. 1, no. 1, pp. 4–27, 1990. 21

[79] B. V. Neida, D. Manicria, and A. Tweed, "An analysis of the energy and cost savings potential of occupancy sensors for commercial lighting systems," *Illuminating Engineering Society*, vol. 30, no. 2, pp. 111–125, 2001. 2

[80] E. Nwafor, A. Campbell, and G. Bloom, "Anomaly-based intrusion detection of iot device sensor data using provenance graphs," in *1st International Workshop on Security and Privacy for the Internet-of-Things*, 2018. 59

[81] C. Nytsch-Geusen, T. Nouidui, A. Holm, and W. Haupt, "A hygrothermal building model based on the object-oriented modeling language modelica," in *Proceedings of the Ninth International IBPSA Conference*, vol. 1, 2005, pp. 867–876. 22

[82] O. T. Ogunsola, L. Song, and G. Wang, "Development and validation of a time-series model for real-time thermal load estimation," *Energy and buildings*, vol. 76, pp. 440–449, 2014. 22, 23

[83] E. S. Olivas, *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques*. IGI Global, 2009. 5

[84] J. L. G. Ortega, L. Han, and N. Bowring, "A novel dynamic hidden semi-markov model (d-hsmm) for occupancy pattern detection from sensor data stream," in *New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on*, IEEE, 2016, pp. 1–5. 16

[85] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. 29

[86] N. Pathak, J. Foulds, N. Roy, N. Banerjee, and R. Robucci, "A bayesian data analytics approach to buildings' thermal parameter estimation," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, ACM, 2019, pp. 89–99. 90

[87] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011. 33, 56

[88] Y. Peng, A. Rysanek, Z. Nagy, and A. Schlüter, "Using machine learning techniques for occupancy-prediction-based cooling control in office buildings," *Applied Energy*, vol. 211, pp. 1343–1358, 2018. 10

[89] N. Perez, P. Riederer, C. Inard, and V. Partenay, "Thermal building modeling adapted to district energy simulation," in *Proceedings of the 14th Conference of International Building Performance Simulation Association*, 2015. 22, 23

[90] J. Petersen *et al.*, "SVM to detect the presence of visitors in a smart home environment," in *Proc. Annual International Conference of Engineering in Medicine and Biology Society*, IEEE, 2012, pp. 5850–5853. 15

[91] M. del Pilar Angeles and A. Espino-Gamez, "Comparison of methods hamming distance, jaro, and monge-elkan," in *International Conference on Advances in Databases, Knowledge, and Data Applications. DBKDA*, 2015. 65

[92] G. Plessis, A. Kaemmerlen, and A. Lindsay, "Buildsyspro: A modelica library for modelling buildings and energy systems," in *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, Linköping University Electronic Press, 2014, pp. 1161–1169. 22

[93] D. Robinson, F. Haldi, P. Leroux, D. Perez, A. Rasheed, and U. Wilke, "Citysim: Comprehensive micro-simulation of resource flows for sustainable urban planning," in *Proceedings of the Eleventh International IBPSA Conference*, 2009, pp. 1083–1090. 22

[94] A. J. Ruiz-Ruiz *et al.*, "Analysis methods for extracting knowledge from large-scale wifi monitoring to inform building facility planning," in *Proc. International Conference on Pervasive Computing and Communications (PerCom)*, IEEE, 2014, pp. 130–138. 11

[95] S. Salakij, N. Yu, S. Paolucci, and P. Antsaklis, "Model-based predictive control for building energy management. i: Energy modeling and optimal control," *Energy and Buildings*, vol. 133, pp. 345–358, 2016. 23, 25

[96] F. C. Sangogboye *et al.*, "Performance comparison of occupancy count estimation and prediction with common versus dedicated sensors for building model predictive control," in *Building Simulation*, Springer, vol. 10, 2017, pp. 829–843. 3, 10, 12, 59

[97] J. Scott *et al.*, "Preheat: Controlling home heating using occupancy prediction," in *Proc. 13th International Conference on Ubiquitous Computing*, ACM, 2011, pp. 281–290. 10

[98] J. Seem, S. Klein, W. Beckman, and J. Mitchell, "Transfer functions for efficient calculation of multidimensional transient heat transfer," *Journal of heat transfer*, vol. 111, no. 1, pp. 5–12, 1989. 26, 86

[99] H. Shi, J. Liu, and Q. Chen, "HVAC precooling optimization for green buildings: An RC-network approach," in *Proc. 9th International Conference on Future Energy Systems (e-Energy)*, ACM, 2018, pp. 249–260. 3, 11

[100] J. Taneja, A. Krioukov, S. Dawson-Haggerty, and D. Culler, "Enabling advanced environmental conditioning with a building application stack," in *Proc. International Green Computing Conference Proceedings*, Jun. 2013, pp. 1–10. 2, 10

[101] The Dark Sky Company LLC, *Dark sky API*, Online `https://darksky.net/dev`, 2018. (visited on 2018). 39

[102] A. Trivedi *et al.*, "iSchedule: Campus-scale HVAC scheduling via mobile WiFi monitoring," in *Proc. 8th International Conference on Future Energy Systems (e-Energy)*, ACM, 2017, pp. 132–142. 11, 76

[103] C. Wang and S. Mahadevan, "Heterogeneous domain adaptation using manifold alignment," in *Proc. 22nd International Joint Conference on Artificial Intelligence - Volume Two*, ser. IJCAI'11, AAAI Press, 2011, pp. 1541–1546. 54

[104] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, 2018. 29

[105] T. Zhang, A. Al Zishan, and O. Ardakanian, "Odtoolkit: A toolkit for building occupancy detection," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, ACM, 2019, pp. 35–46. 7, 70

[106] T. Zhang and O. Ardakanian, "A domain adaptation technique for fine-grained occupancy estimation in commercial buildings," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, ser. IoTDI '19, ACM, 2019, pp. 148–159. 5, 7, 59, 70

[107] D. P. Zhou, Q. Hu, and C. J. Tomlin, "Quantitative comparison of data-driven and physics-based models for commercial building hvac systems," in *American Control Conference (ACC), 2017*, IEEE, 2017, pp. 2900–2906. 3, 11, 22

[108] D. Zhou, Q. Hu, and C. J. Tomlin, "Model comparison of a data-driven and a physical model for simulating hvac systems," *CoRR*, vol. abs/1603.05951, 2016. 3

[109] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint*, 2017. 30

[110] N. Zhu, S. Wang, Z. Ma, and Y. Sun, "Energy performance and optimal control of air-conditioned buildings with envelopes enhanced by phase change materials," *Energy conversion and Management*, vol. 52, no. 10, pp. 3197–3205, 2011. 22, 23

[111] H. Zou *et al.*, "WinLight: A WiFi-based occupancy-driven lighting control system for smart building," *Energy and Buildings*, 2017. 2, 11, 12