Building and Environment 160 (2019) 106177

Contents lists available at ScienceDirect

Building and Environment

journal homepage: www.elsevier.com/locate/buildenv

Comparison of machine learning models for occupancy prediction in residential buildings using connected thermostat data

Brent Huchuk[a,*], Scott Sanner[a], William O'Brien[b]

[a] Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada
[b] Department of Civil and Environmental Engineering, Carleton University, Ottawa, Canada

ARTICLE INFO

Keywords:
Connected thermostats
Residential HVAC
Predictive controls
Occupancy

ABSTRACT

Occupancy detection capabilities provided by modern connected thermostats enable adaptive thermal control of residential buildings. While this adaptation might simply consider the current occupancy state, a more proactive optimized system could also consider the probability of future occupancy in order to balance comfort and energy savings. Because such proactive control relies on accurate occupancy prediction, we comparatively evaluate a number of machine learning models for predicting measurements of the future occupancy state of homes that is critically enabled by thermostat data from real households in ecobee's *Donate Your Data* program. We consider a variety of models including simple heuristic and historical average baselines, traditional machine learning classifiers, and sequential models commonly used for time series prediction. We evaluate the performance of each model according to temporal, behavioural, and computational efficiency characteristics. Our key overall finding is that the random forest algorithm matched or outperformed the other candidate models, had consistently high accuracy predicting over a range of time horizons, and is relatively efficient to train for individual "edge" devices.

1. Introduction

In North America, up to 60% of energy usage in the residential sector is for heating and cooling systems [1,2]. An effective method for generating energy savings for the heating, ventilation, and air conditioning (HVAC) system without large capital costs (e.g., such as upgrading building equipment) or sacrificing occupancy comfort is adjusting temperature setpoints during unoccupied periods [3]. Historically this function has been accomplished using a programmable thermostat, which when programmed correctly, can provide significant savings. For example, previous studies often found more than 30% in energy savings using thermostats with setbacks [4]. Unfortunately, it is well established that the users often do not properly operate programmable thermostats and receive little or negative savings compared to non-programmable thermostat users [5–7].

One approach to deliver savings is to apply a standard default schedule with all thermostats. A previous, and since suspended, Energy Star certification for programmable thermostats [8] mandated a default program that had setback periods included. Unfortunately, the occupancy patterns of an individual home can be highly unique, thus relying on only a default schedule can result in dissatisfied users and minimal savings [9,10]. Today, the occupancy patterns of a home may be more unique than ever with numerous differing lifestyles (e.g., business travel, after school activities, teleworking, etc.). An alternative to reliance on the user-engagement with the schedule or standard default schedules is to customize the thermostat's reaction to prolonged absence based on the individual occupancy patterns in the space [10,11]. To develop these models, the critical requirement is data collection within the space. While this data has generally been available in commercial applications and extensively tested in that setting [3], it has been relatively difficult to collect in residential systems. Fortunately a new class of thermostats, connected thermostats, are now actively collecting occupancy (or occupancy-related) data. In many cases these devices are currently already operating reactively to unexpected changes in the state of home occupancy (i.e., occupied or unoccupied) [7].

To properly manage the complex dynamic relationship of the heating and cooling loads of a building, HVAC control systems should ideally be more than just reactive; they should be predictive over short horizons of a few hours [12–14]. Improper setbacks can cause prolonged occupant discomfort as the recovery from a setback (or setup) in a home can be expected to take less than an hour, but they can vary depending on the weather conditions or the home's thermal characteristics [15]. Predictions should consider these short horizons of a

* Corresponding author.
E-mail address: brent.huchuk@mail.utoronto.ca (B. Huchuk).

https://doi.org/10.1016/j.buildenv.2019.106177
Received 20 March 2019; Received in revised form 31 May 2019; Accepted 4 June 2019
Available online 15 June 2019
0360-1323/

few hours to avoid user discomfort. In addition, predictive systems should be trained and deployed on an individual's unique occupancy data since the occupancy patterns of each residence will be different [10,16]. While previous research has concluded that machine learning is an ideal candidate for the prediction of short-term occupancy patterns [17], this research has often overlooked the actual data availability from devices in the field [3]. The available data in residential buildings is rarely an explicit occupancy state of the home but rather is related to measurements, often motion, which indicate occupancy. Relying on motion in residential buildings has challenges. Residences have relatively low occupant density and numerous rooms that occupants may or may not be in. Occupants also tend to spend numerous hours asleep during which motion is minimal throughout the home. Even with imperfect predictions, it was found that predictive heating controls in homes did not have substantial efficiency differences when compared to predictions made by a perfect oracle [18]. In practice, occupancy data can be sensitive information, so designed methods need to deal with the data privacy of the users [3]. On-board "edge" computing on the residential thermostats themselves — rather than remote computing — helps maintain data privacy and alleviate data security concerns.

In this paper, we compare the performance of numerous baseline and machine learning methods for predicting future motion states in homes using actual connected thermostat data. In addition to comparing various standard sequential and classification techniques, we explore prediction quality versus seasonal effects, time of day, and occupancy behavioural types. Section 2 covers the background literature and limitations of previous research on predicting occupancy in buildings. Section 3 provides details on the implementation of the eight models we investigate in this research and the 100 thermostats that were used to test them. Section 4 presents and discusses the performance of all eight models and provides detailed analysis on the best performing model. Finally Section 5 draws final conclusions and discusses both limitations of the existing study and suggestions for future work.

2. Background

As researchers and commercial products have sought to improve HVAC control systems, the need to understand and model occupancy patterns has been identified as a major contributor to improvements in energy efficiency [16]. In general, more methods and more diverse sensor networks have been considered in commercial building applications [3,10,17]. The application of occupancy modelling is often undertaken in simulation [19,20] or in small field tests [10,13,21]. In both of these situations, researchers are given considerable control over the experiment and data collection. In Shen, Newsham, and Gunay's [3] review of occupancy detection methods in commercial building applications, they noted that many studies did not rely on standard sensor networks and rather required supplemental or alternative sensing methods.

Researchers often have attempted to classify only the current state of occupancy using various sensor types and not predict future occupancy states [21–23]. The investigations monitored environmental conditions (e.g., temperature or humidity), system interactions (e.g., passive infrared (PIR) or door contacts) and timing information. While understanding the current state is useful in many applications such as lighting control, ideal thermal controls require longer prediction horizons. In extending to longer time prediction horizons of multiple timesteps, Markov models are frequently introduced [13,19,20,24].

Li and Dong [24] developed a Markov model and tested on prediction horizons from 15 minutes (min) to up to 24 hours (h) ahead. They concluded their Markov chain was the most effective only in short prediction lengths. This indicates the diminishing predictive power of previous states at longer prediction lengths and motivates an inclusion of multiple conditional features. Many implementations relied on a Markov models with a time-varying condition to the occupancy state transitions [13,19,24]. The inclusion of temporal components has been found to increase accuracy of occupancy models [21]. Based on the prevalence of Markov models in the literature for occupancy prediction, we included them as one of our candidate predictors. However, we do note that simpler methods actually outperformed Markov models in our work as we demonstrate empirically in Section 4.

The demonstrations of occupancy modelling in residential buildings have, similar to commercial buildings, often relied on unique sensor network and unrepresentative data to the commercially available solutions. The 'neural network house' [25] and 'neurothermostat' [26] were an early implementation of a research test home that fully learned how to operate from its users and was intended to anticipate and accommodate the needs of the residents. The 'PreHeat' algorithm [14] compared observed occupancy patterns and predicted future probabilities of occupancy based on the most similar days in the historic information for only five homes. Candanedo, Feldheim, and Deramaix [27] studied the use of hidden Markov models to define occupancy schedules based on temperature, humidity, light, and occupancy measurements. Even in their controlled environment they noted the occupancy patterns were highly varied day-to-day and could be corrupted by situations such as door and window openings or animals in the home. Lu et al. [28] used a combination of available datasets, their own conducted surveys, and data from eight homes they had instrumented with door and motion sensors to develop a heating strategy based on predicted occupancy states. All of these studies were benefited by a commissioning process which allowed them to understand or select sensor placements in the home. Having this level of context is difficult to achieve with a consumer-installed product and highly diverse building stock.

When not relying on custom experimental data, researchers have also had access to atypical levels of service integration. Kleiminger, Mattern, and Santini [18] utilized a cellular telemetry dataset and predicted occupancy for 45 homes as part of a predictive heating system. Comparing a number of methods developed by others, they found an average accuracy of 85% in the best performing methods. Furthermore, they quantified that on their users (and using their data source), that the theoretical maximum limit of predictability would be 90% on average. This level of data sharing between services (i.e., cellular phone, the cellular network, and the HVAC of the home) is often not seen in consumer products and in practice could be subject to privacy concerns.

In general, while the various residential studies often utilized a longitudinal dataset per home similar to those available today from an installed connected thermostat, they remain an idealistic representation of sensing levels and interactions between services. In comparison we rely only on the sensor network of a single sensor type (PIR motion sensors) deployed with the thermostat after installation. Exact placement of sensors and understanding of the coverage of the home is unknown. Similarly, it remains unknown what the ability of an uncommissioned sensor network has to predict the motion states in the home.

3. Methodology

A description of the data source from connected thermostats used in the investigation is described in Section 3.1. Based on the relations and models used by previous researchers, the conditional relations to motion detection are described in Section 3.2. Finally an overview of the selected models that were tested and compared can be found in Section 3.3.

3.1. Data

The *Donate Your Data* (DYD) dataset [29] from connected thermostat maker ecobee Inc. was used to test the various occupancy

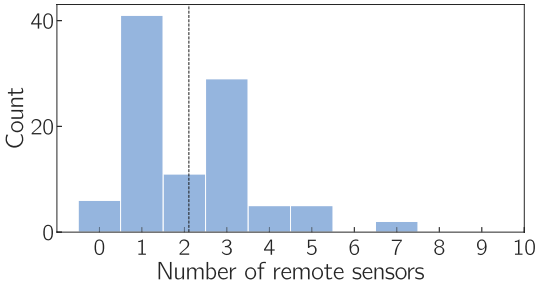

Fig. 1. Distribution of remote sensors, in addition to the on-device PIR sensor, connected with the 100 thermostats used in the sample. The dashed line indicates the average number of remote sensors.

prediction models that we evaluate in this article. A general description of the DYD dataset beyond the subset we use here can be found in other works [30,31]. From the general population of over 25,000 connected thermostats in the dataset, 100 thermostats were selected at random from those that had been online over at least a 12 month period starting in September 2016 and which supported motion sensing. Geographically, the thermostats are located across North America and are predominantly located in single family homes. Given both the time period and thermostat model requirements, the sample group consisted of the ecobee3 thermostat model. The ecobee3 has remote motion detection capabilities with remote sensors in addition to the motion detection on the thermostat itself. The remote sensors each contain a temperature sensor and PIR sensor and communicate using a 915 MHz radio intended to be placed within 45 feet (13.7 meters) of the thermostat [32]. The resulting sample of thermostats had on average over two remote sensors measuring motion in addition to the one on the thermostat. The complete distribution of remote sensors is shown in Fig. 1. The highest number of sensors seen in this group was seven, however the device is able to support up to 32 potential sensors. The effectiveness of the PIR sensors at detecting occupancy in the home is highly dependent on the sensor placement, the number of sensors in the home, the number of people in the home, and the general mobility of occupants throughout the day. It is not possible for the 100 homes selected to identify the effectiveness of all the sensors to capture the true occupancy of the home.

The thermostat interval data, which is reported at 5-min intervals, was processed into 30-min intervals to better align with the schedule blocks on which the thermostat operates and to help reduce noise in the signal. The bins start at the zeroth and 30^{th} min of each hour. While the thermostat schedule operates at this frequency, HVAC equipment runtimes and user overrides are free to operate at rates higher than even the five minute intervals of the data. In Fig. 2, an example of the mapping from the 5-min to 30-min interval data is shown for a two week period for a single thermostat from the sample of 100. The aggregation of the data from Fig. 2a generates Fig. 2b. The occupancy data in Fig. 2b is generated using a logical-OR across all the of the PIR sensors connected to the device. A similar strategy was used by Kleiminger, Mattern, and Santini [18]. In generating Fig. 2b, if all the 5-min observations were missing, then the 30-min observation was also considered missing. If any motion was detected by *any* motion sensor (remote, or on the thermostat itself), the 30-min block was considered as being in the motion state. A period of missing data appears regularly at 18:30 on multiple days in Fig. 2a and b and would appear to be caused by a recurring event (e.g., an Internet connectivity outage) but is not diagnosable with the available data.

3.2. Candidate features for prediction of motion detection

The selection of candidate features that can predict future motion (or occupancy) was guided by the work of other researchers in both the commercial and residential domains outlined in Section 2. For all features that we consider, we indexed time t in 30 min increments so that if t is considered the current time of interest, then $t-1$ refers to 30 min before t and $t-2$ refers to one hour before t. It is considered that the detection of future motion M_t at time t (taking value *true* if motion was detected or *false* otherwise) could depend on the previous state (M_{t-1}) [13,19] and perhaps even states before that (e.g., M_{t-2}, M_{t-3}) since longer periods of motion absence or presence may be more likely to persist. The state M_t could also depend on the time of day H_t as it is noted to be an important contributing factor of occupancy patterns in buildings [19,21,23]. H_t is assigned one of 48 values in {0,1,2...,47} reflecting the index of the 30-min bins of the aggregated data (as shown in Fig. 2b). For example, $H_t = 0$ corresponds to the midnight-12:30am time bin that starts a day while $H_t = 47$ corresponds to the 11:30pm-midnight time bin that is the last in a day. Finally, as it has been observed that occupancy patterns can be highly dependent on whether it is a weekday or weekend [19,21,23], we considered that the state M_t could depend on W_t (taking value *false* on a weekend or *true* on a weekday).

Given occupancy states prior to M_t and knowledge of H_t and W_t at all times t (hour of day and weekday/weekend state are always known), our objective in this paper is to predict occupancy states over the next three hours, i.e., M_t, M_{t+1}, ···, M_{t+5}.While previous model predictive control implementations have used prediction horizons from 15 min to over 24 h [12–14,24], we assume that for the vast majority of *residential* HVAC systems, three hours of future occupancy prediction at 30-min intervals should provide a sufficient time horizon and granularity to make control decisions that could have a significant impact on HVAC efficiency and allow enough time to recover temperature before an occupant may arrive. Nonetheless, all predictive models we consider in this article could be extended to longer or more granular time horizons if desired.

In the training and evaluation of various models, we note that as observed by Page et al. [19], seasonality is another key consideration in predicting occupancy. Because all of our models are intended to be continuously trained on recent data, we addressed the issue of seasonal variation by testing at four different times of the year. The test start dates are shown in Table 1 (right column) and were designed to represent a summer, winter, spring, and fall season. For each of these test cases, the models for each of the 100 thermostats was trained on the previous eight weeks from the test start date. The testing of the models was conducted on the two weeks of data starting on the test start date.

3.3. Models

In our empirical comparison, we consider three groups of models: simple baselines, standard classification models, and sequential models. The baseline models included the most frequent motion state for a residence (independent of anything else) as well as simple models that used only a single feature such as the time of day or the previous occupancy state; they were intended as simple reality checks to ensure that the more sophisticated learning methods were indeed able to learn more complex and better performing predictive models. The classification models predict the motion state at a single time point and hence require training 6 *independent* models for M_t through M_{t+5}. We specifically select logistic regression [33] and random forest [34] because of their widespread use and well-known strong performance on general classification tasks, but we acknowledge other machine learning methods could be used. For example, support vector machines [33] were tested however their performance was ultimately similar to that of logistic regression. In contrast, the sequential models *simultaneously* predict for the full sequence of six motion states (M_t, M_{t+1},...,M_{t+5}) in a single prediction. The sequential models we consider are the Markov model (MM), the hidden Markov model (HMM) [35], and the recurrent neural network (RNN) [36]. The potential advantage of sequential models in general is that they may provide more sequentially coherent predictions than independent classification models; for example,

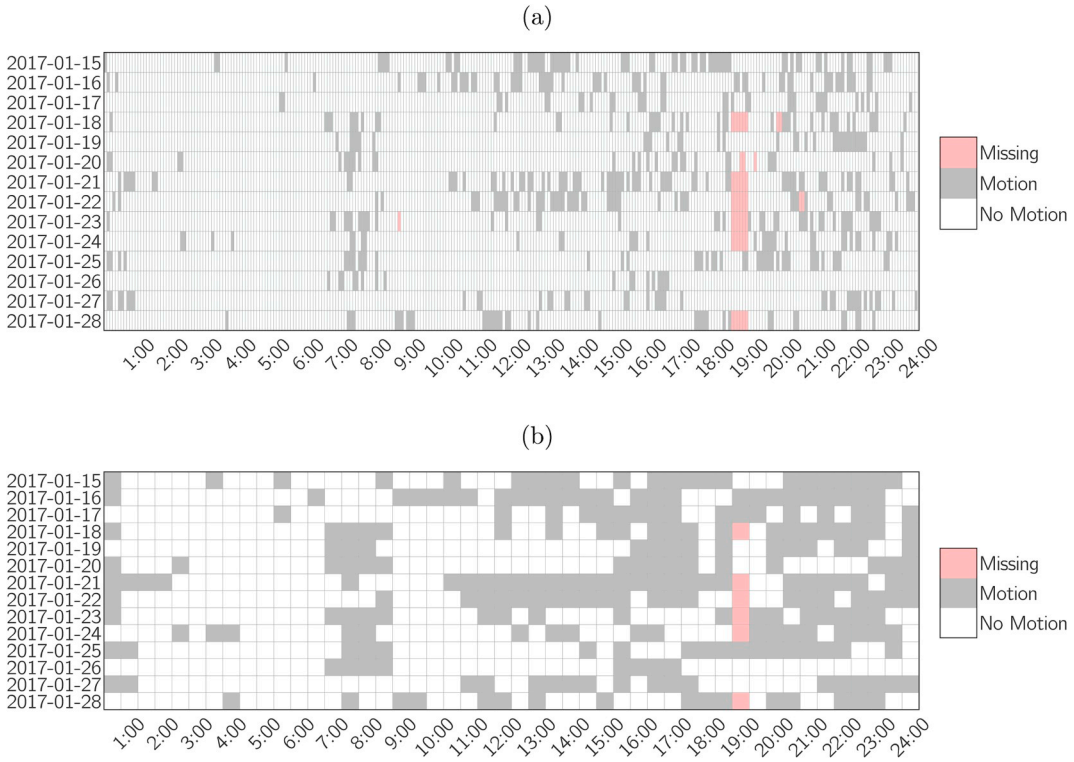

Fig. 2. PIR motion detection from a single thermostat and its remote sensors using (a) 5-min interval data and (b) data mapped to 30-min intervals.

Table 1
Test start date for the four seasonal tests.

Test	Season	Test Start Date
1	Summer	2017-08-06
2	Winter	2017-01-15
3	Spring	2017-05-14
4	Fall	2016-10-16

sequential models may be less likely to predict a transient motion at a given timestep if the timesteps before and after are not predicted to have motion. Unlike MMs, HMMs and RNNs both use a hidden state representation that can learn latent behavioural characteristics of occupants. Compared to the more historically popular HMM, the RNN is a state-of-the-art deep learning method that is often observed to outperform the HMM.

Whether these model differences are actually important for the performance of residential motion prediction with our data and evaluation setting is a question we seek to evaluate empirically. But first we outline each of these models in more technical detail.

3.4. Baseline models

To compare the results of the various testing methods, three simple models were constructed to be used as baselines in comparison. The first model calculated the most frequent state (motion or no motion) during the training period and used this value for all predictions. For example, if a house during the training period was in a no motion state *over* 50% of the time, the predictions were always of a no motion state for any M_t. The second model calculated the most frequent state during a certain time bin H_t over the entire training period for a given day type W_t and used this value for prediction of M_t based on the day type and time bin at time t. The final baseline model for testing copied over the last observed state M_{t-1} to predictions for all of M_t, M_{t+1}, …, M_{t+5}.

3.4.1. Logistic regression

Logistic regression is a commonly used classifier. For the motion prediction task in this article, the logistic model predicts the probability of motion M_t at timestep t conditioned on a vector of features $\mathbf{x}_t$ with length $|\mathbf{x}_t|$ as follows:

$$p(M_t|\mathbf{x}_t) = \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^{|\mathbf{x}_t|} \beta_i \cdot \mathbf{x}_{t_i})}}. \quad (1)$$

Specifically for this paper, we define a unique binary feature $\mathbf{x}_{t_i}$ for every combination of previous occupancy state M_{t-1} (2 possibilities: motion or no motion), time bin H_t (48 bins), and weekend indicator W_t (2 possibilities: weekend or weekday). This leads to a total of $|M_{t-1} \times H_t \times W_t| = 192$ possible combinations leading to a vector length of $|\mathbf{x}_t| = 192$. We specifically use a *one-hot* encoding for $\mathbf{x}_t$ that assigns each of the 192 combinations of feature configurations $M_{t-1} \times H_t \times W_t$ to a unique index i in $\mathbf{x}_t$; hence, in a state at time t with feature configuration i, we set $\mathbf{x}_{t_i} = 1$ and then for all other $j \neq i$, we set $\mathbf{x}_{t_j} = 0$. We use this one-hot encoding for $\mathbf{x}_t$ to allow logistic regression to learn an individual weight β_i for each specific feature configuration i and corresponding binary feature $\mathbf{x}_{t_i}$ as shown in Equation (1). The parameters $\beta_k \in \mathbb{R}$ for $k \in \{0,1,...,192\}$ are learned in order to maximize the conditional likelihood of the training data.

Discriminative classification methods such as logistic regression are widely used since they directly optimize the conditional likelihood of the target classification variable. They are robust to feature dependence (correlation) as opposed to other non-discriminative classification methods such as Naive Bayes. Implementation of the logistic regression classifier in this article was performed using Scikit-learn [37], which exposes an additional hyperparameter constant C corresponding to the inverse of the regularization strength on an L_2 regularizer intended to help prevent overfitting. During the training of each model (one model for each thermostat in each test), the hyperparameter C was tuned using three-fold cross validation by searching for the best value $C \in \{10^{-4}, 10^{-3}, ..., 10^{3}, 10^{4}\}$. Because each classifier is only trained for a single timestep prediction, we trained six *independent* logistic regression models for each thermostat to predict for each future timestep M_t, M_{t+1},

…, M_{t+5}.

3.4.2. Random forest

Random forest is known as an *ensemble* classification technique which trains *multiple* decision trees [34]. In predicting the class label, the decision is based on the mode of the multiple trees as opposed to the result of a single tree. Compared to single decision trees for classification, a random forest is less susceptible to overfitting [34]. Unlike the other tested sequential and classification models, the input features were not encoded – a distinct implementation advantage. By default with the Scikit-learn library [37], the classifier learns 10 trees. Similar to the logistic regression implementation, we used three-fold cross validation to tune the remaining hyperparameters of the random forest for each thermostat's models independently via grid search. In particular, the hyperparameters to tune included the maximum depth limit of the trees (choices: {3, ∞}) and the minimum number of data samples required at a leaf node (choices: {1,3,10}). Similar to the logistic regression classification model, we trained six *independent* random forest models per thermostat to predict for each future timestep M_t, M_{t+1}, …, M_{t+5}.

3.4.3. Markov model

The first and simplest sequential model that we consider is the Markov model depicted as a simple Bayesian network graphical model in Fig. 3. In this model, we assume that the motion state M_t at time t depends only on the previous motion state M_{t-1} as well as the other known values of H_t and W_t. Formally this means that we need only learn the conditional probability $P(M_t|M_{t-1}, H_t, W_t)$ from our data, which for maximizing conditional likelihood in a tabular Markov model corresponds to simple empirical frequency estimates.

Given the learned $P(M_t|M_{t-1}, H_t, W_t)$, we can then instantiate a Markov chain for prediction over future timesteps as shown in Fig. 4 by using this same conditional probability for *all* timesteps. On one hand, this corresponds to a homogeneous Markov model (one whose transition probability does not change with timestep t), but on the other hand, since time H_t and weekday/weekend status W_t are conditioned on for each transition, this could be considered a time-dependent inhomogeneous Markov model as often used in the related work discussed in Section 2.

Once the Markov model transition parameters are learned and the model instantiated as in Fig. 4, the model can be used to predict the motion state for future timesteps. Specifically, we use the pgmpy tool [38] to perform marginal probability inference for each of the following six conditional queries:

$P(M_t|M_{t-1}, H_t, W_t)$

$P(M_{t+1}|M_{t-1}, H_t, W_t, H_{t+1}, W_{t+1})$

⋮

$P(M_{t+5}|M_{t-1}, H_t, W_t, ..., H_{t+5}, W_{t+5})$

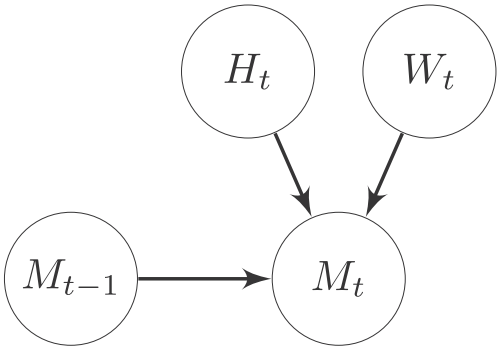

Fig. 3. Graphical model representing the conditional dependencies between current motion (M_t), previous motion (M_{t-1}), hour of the day (H_t), and day type (W_t).

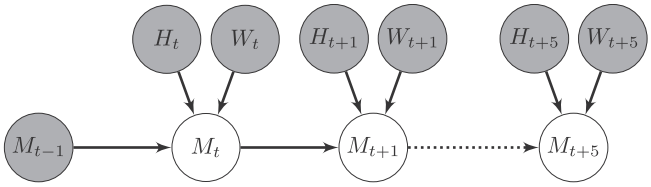

Fig. 4. Schematic of the Markov chain. The shaded nodes are all observed while the unshaded nodes can be queried.

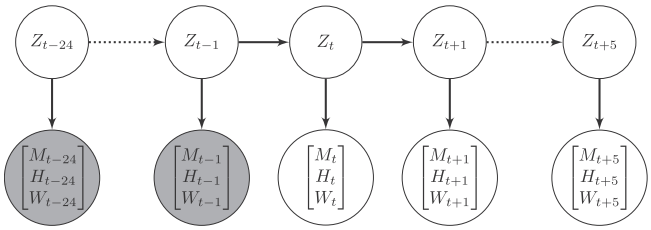

Fig. 5. Representation of states (Z_t) and observations ((M_{t-1}, H_{t-1}, W_{t-1})) in a general hidden Markov model. The shaded cells are observed during training and testing.

We can then use the respective highest probability motion state of each conditional query as the prediction for timesteps M_t, M_{t+1}, …, M_{t+5}. Unlike the classification models which were trained independently per timestep, we only learn and do inference on one Markov model to predict for all future timesteps.

3.4.4. Hidden Markov model

The hidden Markov model (HMM) [35] is a variation of our Markov model (section 3.4.3) depicted in Fig. 5 in which we let (M_t, H_t, W_t) collectively represents the observations at timestep t. A hidden state Z_t at time t is intended to represent latent characteristics of the motion history (e.g., behavioral patterns) that are predictive of future observations. The hidden state Z was chosen to only have two possible values (i.e., occupied or unoccupied). In a test allowing a third potential state for Z, the additional state was not found to improve results.

The HMM was trained on sequences of 24 complete observations from $t-24$ to time $t-1$. In cases of a missing observation, the sequence was rejected. Because the ground truth state is not explicitly observed in the data, training the HMM involves learning both an observation probability and state transition probability function with the expectation maximization (EM) algorithm (also known as Baum-Welch) [35], which is implemented in the HMM Toolbox [39]. We allowed up to 100 EM iterations during training and repeated the entire training procedure 30 times, retaining the highest accuracy model from the 30 training runs.

After training, the HMM can be used for prediction. For prediction, a history of the previous 12 h segments (24 timesteps) was provided. The forward-backward algorithm [35] was used to find the probability of being in a final state (Z_{t-1}) given all the observations (i.e., $P(Z_{t-1}|(M_{t-24}, H_{t-24}, W_{t-24}), ..., (M_{t-1}, H_{t-1}, W_{t-1}))$). Using the translation matrix found during training with EM, the next state (Z_t) was calculated. The Z_t to Z_{t+5} states were calculated sequentially. Using the emission factor, also calculated during EM, the state values are translated to the probability of an observation being made. Given that the hour and weekday value are known, we are able to compare just the two potential observation probabilities representing a motion or no motion value at that time and day-type combination. Of the two observations, the one with the highest probability of being observed is selected and its corresponding motion state taken as the predicted value.

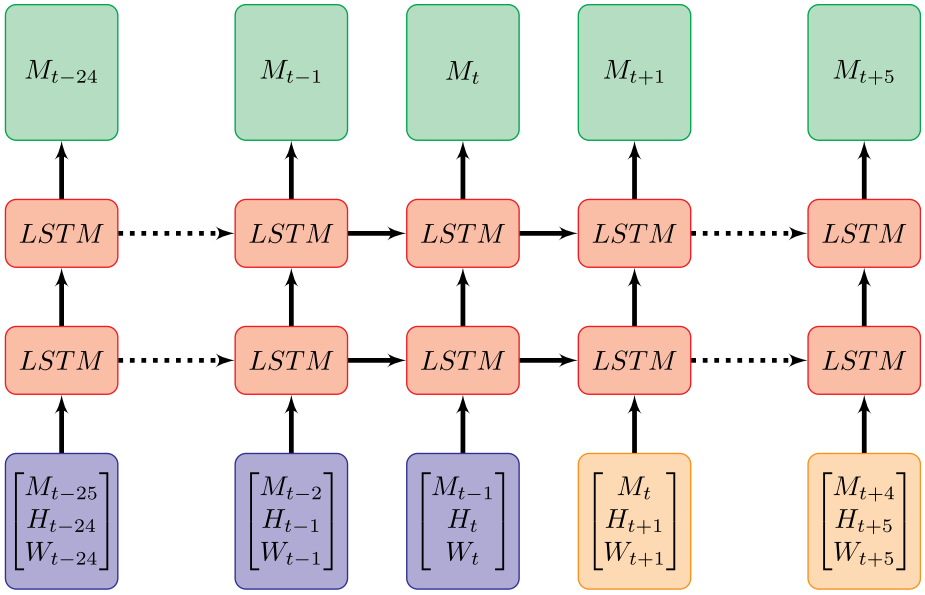

Fig. 6. Schematic of the implemented RNN. The input feature vectors ((M_{t-1}, H_t, W_t)) are passed into two layers of LSTM cells and results in a predicted motion state (M_t). In the orange input vectors, the input feature M_{t-1} had an additional unobserved state value introduced to allow later inference. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

3.4.5. Recurrent neural network

The recurrent neural network (RNN) is a deep learning approach often applied in sequential applications [36]. Shown in Fig. 6, a long short-term memory (LSTM) network [36] provides latent memory of history based on the input combination of features at each timestep. Our features (M_{t-1}, H_t, W_t) collectively represent the observations at timestep *t*. An RNN is capable of learning latent behavioral patterns of residences similar to the function of the latent state in HMMs. RNNs support much more complex architectures than HMMs and do not have a probabilistic internal state. At time of inference it is just efficient feedforward.

Our RNN (Fig. 6) was constructed in Keras [40]. The architecture was set following the tuning of a number of parameters manually though successive runs and monitoring improvements in accuracy during testing. The number of LSTM layers was selected as two (from the choices: {1,2}) and the number of elements to each layer was selected as eight (from the choices: {4,8,16,32}). With each LSTM layer a dropout layer was included to help reduce overfitting. The fraction of input layers to be dropped out was set at 0.2 (selected from the choices {0.1, 0.2, 0.3, 0.4, 0.5}). The size of each batch was varied from {25,50,100} and set to 50. Finally, the number of epochs to train was tuned by monitoring the overfitting during training with five being the chosen number. A complete overview of the finalized architecture can be seen in the Appendix.

The RNN was trained on sequences of 30 complete observations from $t-24$ to time $t+5$. The previous motion state had a new value introduced so that now there was an *unobserved* value. As such the input motion state $M_{t-1} \in \{0,1,2\}$. This new state represented a missing previous state needed later for inference for time t to $t+5$. During both train and testing a motion state (green cells in Fig. 6) was predicted for all timesteps $t-24 \ldots t+5$. All off of the output values were considered during training accuracy, however during our analysis, we only utilized the outputs for the $t \ldots t+5$ timesteps.

4. Results and analysis

To briefly review our evaluation methodology, we compared each of the prediction methods discussed in Section 3 on 100 homes in the ecobee *Donate Your Data* (DYD) dataset as described in Section 3.1. For each of the four seasonal test dates listed in Table 1, we trained the model on the previous eight weeks of data and tested on the following two weeks starting on the specified day. In the following sections, we seek to evaluate the overall accuracy of these methods as well as variation in accuracy as a function of season, time of day, weekday vs. weekend, and simple user behavioral patterns. We also compare the computational requirements of the models and attempt to quantitatively address the relationship between motion detection and occupancy.

4.1. Overall accuracy of methods

We begin our analysis by examining the daily average accuracy for each of the 100 thermostats for all four seasonal test dates. Specifically we define daily average accuracy for a single thermostat as the following expression averaging over all 14 days in the test period and each of the 48 30-min timesteps in a given day:

$$\frac{1}{14} \cdot \frac{1}{48} \sum_{d=1}^{14} \sum_{t=1}^{48} \mathbb{I}\left[\hat{M}_t^d = M_t^d\right], \tag{2}$$

where $\hat{M}_t^d$ is the predicted value by the model in time bin t on the given test day d, M_t^d is the corresponding actual observed motion value in the data, and $\mathbb{I}[\cdot]$ is the indicator function that takes value 1 if its argument is true and 0 otherwise.

In Fig. 7, the distribution of daily average accuracy values for the 100 thermostats is presented as a boxplot for each prediction method and each of the six future prediction timesteps. By analyzing all future prediction timesteps from M_t to M_{t+5}, we can observe whether certain predictors are better over short or longer prediction horizons and the overall difficulty of prediction at each horizon. For the shortest horizon M_t, we make a few key observations: the best median performance is given by a near tie between random forest, logistic regression, and the Markov model with just under 0.80 average accuracy; the previous state baseline also does well here since one would naturally expect adjacent timesteps of motion observation M_{t-1} and M_t to be highly correlated. At the longest prediction horizon of three hours in the future given by M_{t+5}, the best methods still achieve a median accuracy of approximately 0.75 followed closely by the RNN, which appears to perform better for longer horizons than the shortest horizon M_t. The performance of most methods appear to converge to a similar

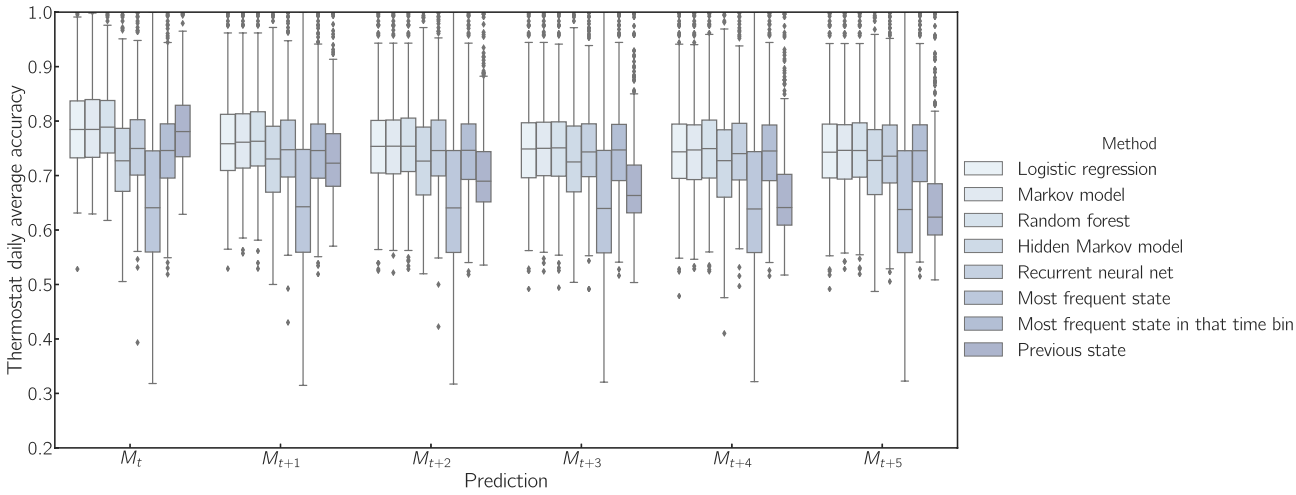

Fig. 7. Thermostat average daily accuracy at the six timestep prediction lengths using the eight proposed models.

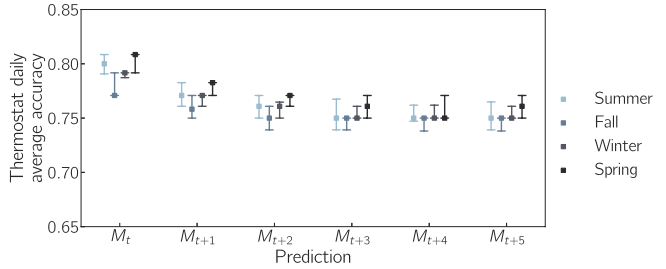

Fig. 8. Median thermostat average daily accuracy at the six timestep prediction lengths with 95% confidence intervals for random forest in each seasonal test case.

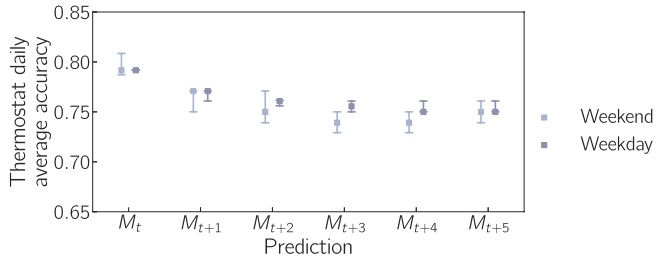

Fig. 9. Median thermostat average daily accuracy at the six timestep prediction lengths with 95% confidence intervals for weekdays and weekends using the random forest algorithm.

performance to the most frequent state in the time bin baseline. As past motion becomes less indicative of future motion for prediction, the frequentist approach to prediction based on time of day and day type should become our best estimation. We observe that random forest generally provides improved median daily average accuracy across all time prediction horizons with 1st and 3rd quartile accuracies also better than or comparable to the best performing methods. This indicates that random forest provides reliably strong performance across the range of thermostats in the evaluation though not substantially better than the logistic regression or the Markov model. Overall, while our sample size was limited to 100 thermostats, we believe the upcoming analysis in Figs. 8–11 demonstrates a range of diverse occupancy behaviours in our sample and hence increasing the sample size beyond 100 is unlikely to appreciably increase the diversity or significantly change the ranking of the top-performing models.

In comparing these results to previously published results in different occupancy prediction settings, other studies have also found accuracy values around 0.80 to 0.85 [3,18]. The studies achieving higher accuracy could be a result of utilizing a different sensing method, having more predictable occupancy patterns or higher ratios of sensing to occupants. In a sample of homes, it was found that the theoretical limit of predictability may only be 0.90 [18] owing to natural noise in the sensing process. That baseline however was using an extremely accurate sensing technology in cellular network triangulation and different movement patterns which would change the theoretical maximums in our study.

4.2. Seasonal effects

Based on the observations by Page et al. [19], seasonality was expected to impact occupancy patterns, hence we postulated there may be seasonal variation in prediction accuracy if some seasons had more stochasticity in occupancy than others. Fig. 8 shows the same daily average accuracy for each thermostat as computed in Equation (2), but grouped by season and only for the top method, random forest. At each prediction horizon, the seasonal performances are generally statistically indistinguishable given the 95% confidence intervals. While not statistically significant, the data weakly suggests that Spring is the most predictable season while Fall appears to be harder to predict for short time horizons. Overall, the results generally suggest that while there may be some seasonal variation, motion can be predicted in all seasons using random forest with a fairly high accuracy (i.e., >0.75).

4.3. Results by day type

In Fig. 9, the median average thermostat daily error values with 95% confidence intervals are shown for all prediction horizons for both weekdays and weekends. As with seasons, the weekday vs. weekend performance are generally statistically indistinguishable given the 95% confidence intervals. While not statistically significant, the data weakly suggests that it may be slightly easier to predict for some middle time horizons for weekdays than weekends. This could indicate people's schedules on weekdays are more predictable (e.g., given a weekday work schedule) than on the weekend when schedules may be more varied.

4.4. Results by time of day

We now wish to investigate how model prediction accuracy varies over the day. In Fig. 10, we show a discrete heatmap for the median average accuracy across all thermostats using the various models versus the time of the day. We only show predictions made for the time period as a prediction at the two time horizon extremes of (a) M_t and (b) M_{t+5}.

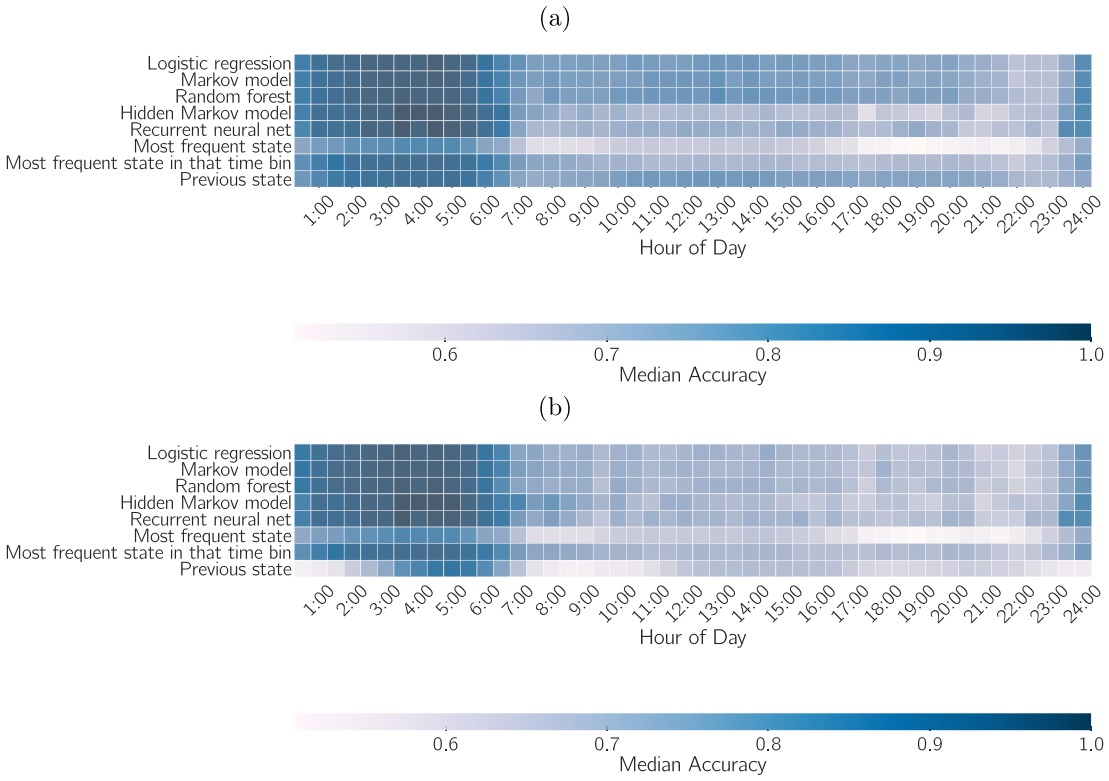

Fig. 10. Thermostat average daily accuracy for the various methods for predictions made for that time of day as a prediction of (a) M_t and (b) M_{t+5}.

For example, the 9:00 bin is for predictions of that time made at either 8:30 for M_t or 6:00 for M_{t+5}. The intensity of the shade implies a higher median accuracy value. The logistic regression, Markov model, and random forest are the best performing models and exhibit similar temporal patterns of prediction accuracy across the day. In contrast the hidden Markov model and RNN appear more similar to the baseline using the most frequent state in that time bin.

Unsurprisingly, the highest accuracies are found during the middle of the night (1:00 to 6:00) when most residences have little motion detected. All models consistently appear to have the most difficulty in the late evening (around 22:00) for predicting M_t; we conjecture that while households tend to wake up reliably around the same time every day, exact bedtimes may be less predictable. For the longer horizon (three hours ahead) prediction of M_{t+5}, average predictive accuracy is lower than for M_t as expected, but with increased uncertainty starting around 23:00 persisting until early into the morning. Overall, while predictive accuracy for the best performing methods is lowest in the evening period up until bedtime, these may be the riskiest times for deep setbacks; hence this observed reduction in prediction accuracy may not have a major impact on practical on thermostat controls that may only attempt deep setbacks in the more predictable earlier parts of the day.

4.5. Results by user type

Now we wish to assess prediction accuracy over a range of user behavioural types, namely the average amount of motion sensed by a thermostat in a household. In Fig. 11, the median daily average accuracy per prediction model is plotted in a discrete heatmap for different ranges of user types in columns; the median is taken over all thermostats satisfying the user type conditions. The user type was quantified as the fraction of time that a thermostat detected a positive motion state during the eight week training period. For example, the thermostat for a user type of 0.6 would indicate that an active motion state is detected on average 60% of the time periods in a day. We remark that no homes in the selected 100 thermostats showed an active motion state higher than 70% of the time.

Fig. 11a shows the 30 min time horizon prediction for M_t while Fig. 11b shows the three hour time horizon prediction for M_{t+5}. In both figures, higher accuracy is attained by all methods for users with either a large or small fraction of active motion (i.e., in the (0.0, 0.1] and (0.6, 0.7] ranges). The random forest, Markov model, and logistic regression are consistent across all ranges of user type, with the range (0.3, 0.5] showing the lowest accuracy owing to the high uncertainty of motion detection for users in this range. As for the time of day analysis, the baseline method of previous state performs well for M_t in Fig. 11a, but poorly for M_{t+5} in Fig. 11b. The most frequent state method does poorly for both short and long prediction horizons for user types greater than 0.3, since it lacks the context of time. Overall, while it is harder for all models to predict in the (0.3, 0.5] range of user types, this analysis by user type does not reveal the weakness of some of the methods (e.g., HMM, RNN, most frequent state in that time bin) as clearly evidenced by previous analyses.

4.6. Model implementation requirements

Next we examine the computational requirements for each prediction model to determine their ability to be implemented on the low-level hardware at the “edge” of the computing system (i.e., on the thermostat itself). This analysis is shown in Table 2 with training and inference (prediction) times displayed for a single thermostat. The training time was estimated for training a single thermostat during a single seasonal test. Included in the training time was hyperparameter estimation (when done online for each thermostat) and multiple iterations during training such as the five epochs with the RNN or 30 iterations of the HMM. The time for inference was the time to predict for a single timestep. All timings are reported for the same laptop computer with a 2.3 GHz Intel Core i5 processor and 16 GB of RAM.

All machine learning methods expose additional tuning parameters called hyperparameters that can be adjusted to improve learning and

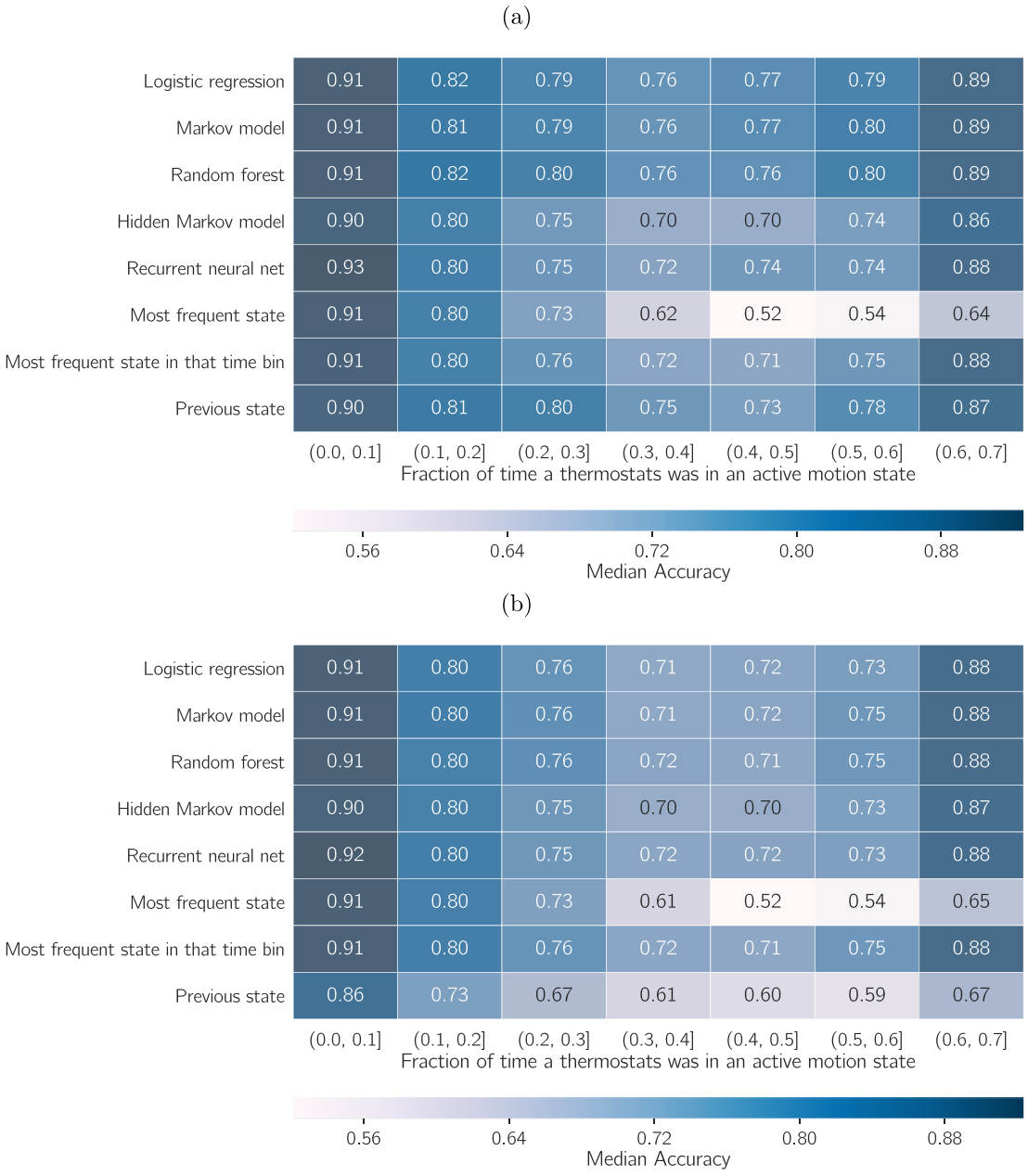

Fig. 11. The median value of thermostat average daily accuracy for the various methods at the (a) M_t timestep and the (b) M_{t+5} timestep given the average daily fraction of the day with detected motion.

Table 2
Estimated inference and training times for the proposed models.

Method	Training Time (s)	Inference Time (s)
Logistic regression	0.16	0.0005
Markov model	0.20	0.001
Random forest	0.38	0.001
Hidden Markov model	240	0.005
Recurrent neural network	10	0.01

generalization quality. In the timing results above, we only showed times for the best global hyperparameters but did include the time to tune parameters for an individual thermostat model. However, it is critical to point out that it is inherently much easier (and faster) to tune hyperparameters for some predictors than others. For example, logistic regression typically only exposes a single hyperparameter (i.e., regularization strength), whereas at the other extreme, RNNs have a large number of hyperparameters (number of layers and hidden units, settings for dropout if used) whose adjustment can drastically affect both learning performance and training time.

When considering the additional overhead of hyperparameter tuning, we remark that HMMs and RNNs required some of the most complex tuning compared to the more easily tuned logistic regression, Markov model, and random forest. Hence in reality, the training times for the HMM and RNN would be even larger than shown. However, these deficiencies of the HMM and RNN are all to some extent a moot point given the performance of the three other machine learning methods. Notably, random forest performed best across all settings in the previous evaluation discussion and is among the most computationally efficient methods for both training and evaluation. While "edge" hardware on the thermostat would be considerably less powerful than the laptop used in the evaluations, at even 100x slower, random forest could still be trained in under a minute and predictions still made in a fraction of a second. If the memory were also extremely

limited, we remark that logistic regression and the Markov model might be preferred over random forest. That is, both logistic regression and the Markov model yielded high accuracy and fast inference and training times comparable to random forest, and furthermore each require learning approximately 192 parameters requiring <1 kB of memory, which is more than an order of magnitude smaller than the typical memory footprint of random forest. We also remark that due to their simplicity, the learned models of both logistic regression and the Markov model can be easily inspected and interpreted for diagnostic purposes.

4.7. Effects of the number of remote sensors

Given that we only had access to data from PIR motion sensors and the existence of ground truth occupancy is not available for the DYD dataset, it is difficult to completely assess the effectiveness of the PIR data and the models at predicting occupancy. In an attempt to establish upper bounds on the effectiveness of using PIR for detecting motion, we looked at the relative increase in the motion detection rate as additional remote PIR sensors were used in a given residence. Of the 25,000 thermostats in the full DYD dataset, we examined the subset of 4,400 thermostats with both PIR motion sensing on the thermostat and three additional remote PIR sensors and calculated the average number of motion detections per day per thermostat. Fig. 12 shows the median value as more sensors were added along with 95% confidence intervals. The results show that the addition of one sensor increases detection by almost 30%. After this, there appears to be diminishing returns with each additional sensor added. The difference between zero and three additional remote sensors is almost a 40% increase in detections. As sensors continue to add some values, we acknowledge our sensor networks are often missing events, meaning our possible accuracy is less than 100%. The theoretical limit of accuracy below 100% has similarly been identified by other researchers [18]. Unfortunately, we are unable to calculate the magnitude of false positive or false negative events by properly comparing occupancy to motion because of a lack of ground truth data.

We remark that the previous analysis does not account for potential home-specific effects such as the size of the home, number of floors, or number of occupants in the home. Nor does it address sensor specific concerns such as remote sensor placement (e.g., facing open space where normal paths are and not facing a wall or furniture) and activity type (sleeping versus active times). These variables could all affect the capability of the remote sensors to accurately detect motion in the home. Despite these potential caveats, we believe the one key take-home point of Fig. 12 is that if we consider motion to serve as a surrogate for occupancy during waking hours, then there would be limited value in terms of increased motion detection of having more than one on-thermostat PIR motion sensor and three additional remote PIR sensors. Furthermore, four PIR sensors is not unreasonable given the thousands of residences that already have this many sensors in the DYD dataset.

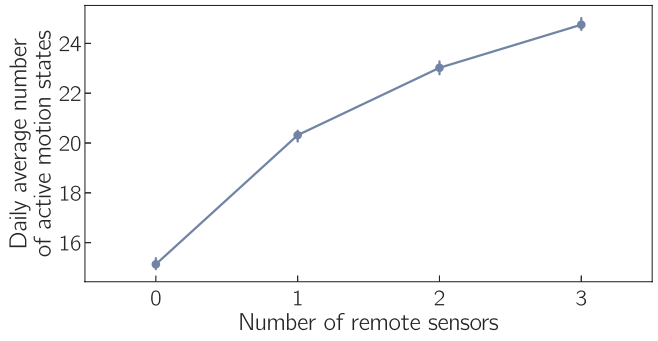

Fig. 12. Average number of daily motion states based on the number of remote senors used across the home on a sample of thermostats with three remote sensors. The 95% confidence intervals are shown.

5. Conclusions and future works

New generations of "smart" residential thermostats attempt to adapt HVAC controls based on changes of occupancy state of the home. In order to be most effective in managing the delayed thermal effects of the home, these thermostats need to go beyond reacting to the current occupancy state and instead predict future occupancy state. Using ecobee's *Donate Your Data* dataset of connected thermostats having multiple PIR motion sensors as a proxy for occupancy, we evaluated a wide variety of occupancy prediction models to evaluate which are the most accurate and computationally viable for implementation on the "edge" (i.e., on the thermostat itself). We also considered how many motion sensors would be needed in a residence and found that improvements in motion detection after installing four PIR sensors (one on the thermostat and three remote) would not be substantial.

Using a set of candidate features consisting of previous motion states, time of day, and weekday vs. weekend status, we found a simple random forest machine learning model to slightly outperform logistic regression and a Markov model and to significantly outperform simple baselines as well as Hidden Markov Models (HMMs) and Recurrent Neural Networks (RNNs). We also found the best models (random forest, logistic regression, and Markov model) to be robust to a range of conditions including the time of day, weekday versus weekends, and for homes with differing occupancy rates. The random forest model had the additional benefit of being implemented without feature encoding, meaning it was able to identify those more complex relations without explicit feature engineering. Computationally, we observed random forest, logistic regression, and Markov model to be most efficient in terms of training and prediction time with logistic regression and Markov model offering the smallest memory footprint as well as interpretability that could support diagnostic purposes. Given the relatively small feature space, it is postulated that the effectiveness of the logistic regression and Markov model could be replicated by a lookup table generated on device. However, with the inclusion of any additional features that would no longer be the case.

Future directions of this work will seek to identify or collect datasets that contain similar logged data for motion sensors but additionally with ground-truth for occupancy and alternative occupancy proxies. Further, we aim to integrate the prediction strategies developed in this paper with control strategies to quantify achievable energy savings with occupancy prediction. At that time, more tuning and model analysis may be required to differentiate false positive and false negative occupancy detection errors as both have different consequences. For example, a false positive (occupancy predicted true when actually false) may have an energy penalty due to unnecessary HVAC use; however, a false negative (occupancy predicted false when actually true) will lead to occupant discomfort if thermal controls are reduced due to this error. Finally, other deep learning methods are continuously being improved and may provide additional insights, particularly with the inclusion of additional (sensed) features. While our analysis was limited to data up-sampled to 30-min intervals, this could for some users actually cause energy increases [41]. This potential issue should be better quantified in order to determine an appropriate temporal granularity for occupancy prediction to enable optimal HVAC control and energy savings.

Due to NDA restrictions, we cannot directly share the data used in this article, but readers are freely able to apply to ecobee for access to the *Donate Your Data* dataset that we used. The code used for this analysis is freely available in the repository: https://github.com/chuck-b/ml_occupancy_model_comparison.

Acknowledgements

This work is funded by a research grant provided by the Natural

Sciences and Engineering Research Council of Canada (NSERC) and ecobee Inc. Two of the authors are active participants of International Energy Agency Energy in Buildings and Communities (IEA-EBC) Programme Annex 79.

The authors wish to make it known that The Corresponding Author has previously been a full-time employee and remains an employee in a diminished role for the organization (ecobee Inc.) while pursuing his studies. The organization provided financial support to the research and do administer the open dataset used in the investigation.

Appendix

The final architecture of the RNN is shown in Fig. 13. The summary, a generated output from Keras [40], shows the particular layer types and their dimensions. Read top down, the LSTM units are followed each time by a single dropout layer. The time distributed layer applies a dense layer to each element of the sequence ahead of the activation layer. Our activation layer used a sigmoid function.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 30, 8)             9504
_________________________________________________________________
dropout (Dropout)            (None, 30, 8)             0
_________________________________________________________________
lstm_1 (LSTM)                (None, 30, 8)             544
_________________________________________________________________
dropout_1 (Dropout)          (None, 30, 8)             0
_________________________________________________________________
time_distributed (TimeDistri (None, 30, 1)             9
_________________________________________________________________
activation (Activation)      (None, 30, 1)             0
=================================================================
Total params: 10,057
Trainable params: 10,057
Non-trainable params: 0
_________________________________________________________________
```

Fig. 13. Keras output showing the final layer structures and shapes for our RNN model.

References

[1] Energy Information Administration, Residential Energy Consumption Survey (RECS), (2015).
[2] NRCan (Natural Resources Canada), Energy use data handbook tables (Canada), http://oee.nrcan.gc.ca/corporate/statistics/neud/dpa/menus/trends/handbook/tables.cfm, (2016).
[3] W. Shen, G. Newsham, B. Gunay, Leveraging Existing Occupancy-Related Data for Optimal Control of Commercial Office Buildings: A Review, (Aug 2017), https://doi.org/10.1016/j.aei.2016.12.008.
[4] J.W. Moon, S.H. Han, Thermostat strategies impact on energy consumption in residential buildings, Energy Build. 43 (2–3) (2011) 338–346, https://doi.org/10.1016/j.enbuild.2010.09.024.
[5] M.J. Nevius, S. Pigg, Programmable thermostats that go berserk? Taking a social perspective on space heating in Wisconsin, Proc. ACEEE Summer Stud. Energy Effic. Build. 8 (2000) 233–244.
[6] M. Pritoni, A.K. Meier, C. Aragon, D. Perry, T. Peffer, Energy efficiency and the misuse of programmable thermostats: the effectiveness of crowdsourcing for understanding household behavior, Energy Res. Soc. Sci. 8 (2015) 190–197, https://doi.org/10.1016/j.erss.2015.06.002.
[7] T. Peffer, M. Pritoni, A. Meier, C. Aragon, D. Perry, How people use thermostats in homes: a review, Build. Environ. 46 (12) (2011) 2529–2541, https://doi.org/10.1016/j.buildenv.2011.06.002.
[8] U.S. Environmental Protection Agency, Programmable Thermostats Suspension Memo, (2009) https://www.energystar.gov/ia/partners/prod_development/revisions/downloads/thermostats/Spec_Suspension_Memo_May2009.pdf.
[9] B. Urban, C. Gomez, A Case for Thermostat User Models, Proceedings of BS2013: 13th Conference of International Building Performance Simulation Association, Chambery, France, August 26-28.
[10] F. Nägele, T. Kasper, B. Girod, Turning up the Heat on Obsolete Thermostats: A Simulation-Based Comparison of Intelligent Control Approaches for Residential Heating Systems, (Aug 2017), https://doi.org/10.1016/j.rser.2016.11.112.
[11] H. Burak Gunay, W. O'Brien, I. Beausoleil-Morrison, Development of an occupancy learning algorithm for terminal heating and cooling units, Build. Environ. 93 (P2) (2015) 71–85, https://doi.org/10.1016/j.buildenv.2015.06.009.
[12] A.A. Panagopoulos, M. Alam, A. Rogers, N.R. Jennings, Adaheat: a general adaptive intelligent agent for domestic heating control, Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, 2015, pp. 1295–1303.
[13] J.R. Dobbs, B.M. Hencey, Model predictive HVAC control with online occupancy model, Energy Build. 82 (2014) 675–684, https://doi.org/10.1016/j.enbuild.2014.07.051.
[14] J. Scott, A. Bernheim Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges, N. Villar, Preheat Controlling home heating using occupancy prediction, UbiComp '11 Proceedings of the 13th International Conference on Ubiquitous Computing, 2011, p. 281, , https://doi.org/10.1145/2030112.2030151.
[15] M.M. Manning, M.C. Swinton, F. Szadkowski, J. Gusdorf, K. Ruest, The effects of thermostat setback and setup on seasonal energy consumption at the CCHT research facility, ASHRAE Trans. 113 PART 1 (1) (2005) 630–641 http://doi.org/10.4224/20378071.
[16] F. Oldewurtel, D. Sturzenegger, M. Morari, Importance of occupancy information for building climate control, Appl. Energy 101 (2013) 521–532, https://doi.org/10.1016/j.apenergy.2012.06.014.
[17] Z. Chen, C. Jiang, L. Xie, Building occupancy estimation and detection: a review, Energy Build. 169 (2018) 260–270, https://doi.org/10.1016/j.enbuild.2018.03.084.
[18] W. Kleiminger, F. Mattern, S. Santini, Predicting household occupancy for smart heating control: a comparative performance analysis of state-of-the-art approaches, Energy Build. 85 (2014) 493–505, https://doi.org/10.1016/j.enbuild.2014.09.046.
[19] J. Page, D. Robinson, N. Morel, J.L. Scartezzini, A generalised stochastic model for the simulation of occupant presence, Energy Build. 40 (2) (2008) 83–98, https://doi.org/10.1016/j.enbuild.2007.01.018.
[20] A.E. Mady, G.M. Provan, C. Ryan, K.N. Brown, Stochastic model predictive controller for the integration of building use and temperature regulation, Proceedings of the 25th AAAI Conference on Artificial Intelligence, 2011, pp. 1371–1376, , https://doi.org/10.1021/jm9804477.
[21] L.M. Candanedo, V. Feldheim, Accurate occupancy detection of an office room from light, temperature, humidity and CO_2 measurements using statistical learning models, Energy Build. 112 (2016) 28–39, https://doi.org/10.1016/j.enbuild.2015.11.071.
[22] R.H. Dodier, G.P. Henze, D.K. Tiller, X. Guo, Building occupancy detection through sensor belief networks, Energy Build. 38 (9) (2006) 1033–1043, https://doi.org/10.1016/j.enbuild.2005.12.001.
[23] Y. Zhao, W. Zeiler, G. Boxem, T. Labeodan, Virtual occupancy sensors for real-time occupancy information in buildings, Build. Environ. 93 (P2) (2015) 9–20, https://doi.org/10.1016/j.buildenv.2015.06.019.
[24] Z. Li, B. Dong, Short term predictions of occupancy in commercial buildings – performance analysis for stochastic models and machine learning approaches, Energy Build. 158 (2018) 268–281, https://doi.org/10.1016/j.enbuild.2017.09.052.
[25] M.C. Mozer, The neural network house: an environment that adapts to its inhabitants, Proc. AAAI Spring Symp. Intelligent Environments, vol 58, 1998, pp. 110–114.
[26] M.C. Mozer, L. Vidmar, R.H. Dodier, The neurothermostat: predictive optimal control of residential heating systems, Adv. Neural Inf. Process. Syst. 9 9 (1997) 953–959 doi:10.1.1.49.2620.
[27] L.M. Candanedo, V. Feldheim, D. Deramaix, A methodology based on Hidden Markov Models for occupancy detection and a case study in a low energy residential building, Energy Build. 148 (2017) 327–341, https://doi.org/10.1016/j.enbuild.2017.05.031.
[28] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, K. Whitehouse, The smart thermostat: using occupancy sensors to save energy in homes, Proceedings of ACM SenSys, vol 55, 2010, pp. 211–224, , https://doi.org/

10.1145/1869983.1870005.
[29] Ecobee Inc, Donate Your Data, (2017) https://www.ecobee.com/donateyourdata/.
[30] B. Huchuk, W. O'Brien, S. Sanner, A longitudinal study of thermostat behaviors based on climate, seasonal, and energy price considerations using connected thermostat data, Build. Environ. 139 (2018) 199–210 https://doi.org/10.1016/j.buildenv.2018.05.003.
[31] M.F. Touchie, J.A. Siegel, Residential hvac runtime from smart thermostats: characterization, comparison, and impacts, Indoor Air 28 (6) (2018) 905–915, https://doi.org/10.1111/ina.12496.
[32] Ecobee Inc., Room Sensor Installation Guide, (2015) www.ecobee.com.
[33] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, C.-J. Lin, LIBLINEAR: a library for large linear classification, J. Mach. Learn. Res. 9 (2008) 1871–1874.
[34] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32, https://doi.org/10.1023/A:1010933404324.
[35] L.R. Rabiner, B.H. Juang, An introduction to hidden Markov models, IEEE ASSP Mag. 3 (1) (1986) 4–16, https://doi.org/10.1109/MASSP.1986.1165342.
[36] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735.
[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[38] A. Ankan, A. Panda, Probabilistic graphical models using python, Proceedings of the 14th Python in Science ConferencearXiv:vol 1011.1669v3.
[39] K. Murphy, Hidden Markov Model (HMM) Toolbox for Matlab, (1998) https://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html.
[40] Keras, Keras Documentation, (2017) https://keras.io/.
[41] S. Gilani, W. O'Brien, J.S. Carrizo, Interpreting occupant-building interactions for improved office building design and operation, ASHRAE Transact. 123 (2) (2017) 185–202.