An uncertainty-aware deep reinforcement learning framework for residential air conditioning energy management

- This paper describes a data-driven framework for uncertainty-aware residential air conditioning control.
- Bayesian Convolutional Neural Networks (BCNN) are utilized to model air conditioning and room temperature dynamics such the random nature of air conditioning components can taken into account for planning.
- Q-learning reinforcement agents are subsequently developed for automated air conditioning control, by taking into account uncertainty from the generated models and external weather conditions.

An Uncertainty-Aware Deep Reinforcement Learning Framework for Residential Air Conditioning Energy Management

Clement Lork[a], Wen-Tai Li[a], Yan Qin[a], Yuren Zhou[a], Chau Yuen[a], Wayes Tushar[b], Tapan K. Saha[b]

[a]*Engineering Product Development, Singapore University of Technology and Design, 8 Somapah Road, Singapore*
[b]*School of Information Technology and Electrical Engineering, University of Queensland, Australia*

Abstract

Most existing methods for controlling the energy consumption of air conditioning (AC), focus on either scheduling the switching (on/off) of compressors or optimizing the overall energy consumption of AC system of an entire building. Unlike commercial buildings, residential apartments typically house separate ACs in individual rooms occupied by people with different thermal comfort preferences. Fortunately, the advancement of Internet-of-Things (IoT) technology has enabled the exploitation of sensory data to intelligently control the set-point temperature of ACs in individual rooms based on environmental conditions and occupant's preferences, improving the energy efficiency of residential buildings. Indeed, control decisions based on sensory data may suffer from uncertainties due to error in data measurement and contribute to model uncertainty. This work proposes a data-driven uncertainty-aware approach to control split-type inverter ACs of residential buildings. First, information from similar AC and residential units are aggregated to reduce data imbalances, and Bayesian-Convolutional-Neural-Networks (BCNNs) are utilized to model the performance and uncertainty of the ACs from the aggregated data. Second, a Q-learning based reinforcement learning algorithm for set-point decision making is designed for setpoint optimization with transitions sampled from the BCNN models. Third, a case study is simulated based on such a framework to show that the control actions taken by the uncertainty-aware agent perform better in terms of discomfort management and energy savings compared to the uncertainty unaware agent. Further, the agent could also be adjusted to capture the trade-off between energy savings and comfort levels for varying degrees of energy and discomfort savings.

Keywords: Bayesian Neural Networks, Air Conditioning, Energy Saving

---

Air conditioning (AC) units within residential buildings account for up to 40% of the total electricity consumption in Singapore and up to 45% of that in the USA [1]. According to [2], there is a gap between actual and theoretical performances of ACs due to over-generalization in simulation and occupancy models. With the advancement of Internet-of-Things (IoT), controls of AC could be implemented more efficiently by considering various sensory data [3] for different target areas to reduce this gap. For example, [4] combined readings from cameras, indoor temperature sensors and outdoor sensors to control the AC of a mosque, with processing being done on a Raspberry Pi. By specifically controlling the setpoints of AC according to different indoor conditions, [5, 6] show that it is possible to reduce the energy consumption of ACs while maintaining human comfort at an acceptable level. However, most residents do not have the time or knowledge to personally optimize the settings of their AC [7], which has motivated the emergence of automated AC control. Leveraging on sensors, computing and IoT technology, many startups have sprung up to fill this market demand. A well known example will be the Google Nest Thermostat, where it combines indoor temperature readings, outdoor temperature forecasts, and user defined schedules to enable energy savings for AC systems.

Existing research on automated AC control can be classified into three types: rule-based, model-free, and model-based. Rule based controllers, such as in [5] and [8], are heuristics to control the on and off time of the AC and other possible actions. Although such rule-based controllers can be simple to implement and provide good energy savings, they are system-specific and might have to redesign if there are changes to the system, for example, a change in room layout. Model-free controllers attempt to learn the optimal policy from direct or historical interaction with the AC environment without special consideration of the dynamics of the system. An example is [9] where a reinforcement learning algorithm is used to learn an optimal policy for AC control in data centers. When used in their vanilla form [10], model-free methods incur huge overheads in terms of data collection as they require a large number of system transitions to learn and converge to the optimal policy. Consequently, model-free methods are not practical in data-scarce scenarios.

Finally, model-based methods are used to capture the dynamics of a system before applying optimization algorithms to decide on the control actions. Model-based methods can further be classified into three kinds: the black-box model, the white-box model, and the grey-box model. White-box models attempt to reconstruct the physical interaction between the AC components and the thermal characteristics of the residence [11]. This approach relies heavily on obtaining the exact measurements of each thermal characteristics such as furniture within rooms and occupancy behavior that are difficult to obtain in real scenarios. Grey-box models estimate the thermal characteristics of the room and the AC components by fitting metered load data to a generic physical model [12]. Black-box data-driven techniques have advantages in modelling the more complex and volatile AC systems as compared to white-box and grey-box techniques, striking a balance between complexity of implementation and accuracy. By not having a pre-defined physical model, black-box models are not restricted to the data requirements of the actual physical model, but are able to predict the AC consumption by mapping interactions between the load data collected and other exogenous information like weather and consumer behaviour with good accuracy. This way, black-box models facilitates model building without the need of physical knowledge on the room (e.g. window size, build material etc.) In recent years, black-box models have received increasing attention to drive automation and reduce human intervention. Examples of black-box models for AC load forecasting includes a regression tree based model [13], a support vector regression model [14], and neural network model [15]. [16] utilized a support vector regression to predict the state of a HVAC system before using a rule-based approach to apply cooling to a zone. [17] applied a mixed logical dynamical model to capture dynamics of a residential HVAC system coupled with a photovoltaic setup, before applying a model predictive control logic for efficient energy usage.

Clearly, existing literature provides valuable insights into how to model various AC loads. In the case when studies do not have the luxury of curating data collection, data imbalance might become an obstacle [18]. Classes of interest are drown-out by other classes, skewing prediction results towards the classes with more data. Further, accuracy of data-driven models decreases when training data deviates from testing data. Although it is ideal for data-driven models to be trained on datasets that cover all operating conditions, gathering a completed dataset could be challenging in real-world scenarios as operating conditions are usually fixed due to habit or operational requirements (unless explored forcibly).

Now, existing models typically employ point forecasts that provide only the mean values for the output. However, when training a model on an incomplete dataset, it is important that the model is able to estimate an uncertainty bound on its prediction so that an optimization algorithm can disregard an uncertain prediction from an unfamiliar scenario in the planning stage. As such, probabilistic modeling has started to gain more interest in recent years. For example, the models proposed in [19] and [20] model aggregated electricity loads with a pinball loss guided LSTM, and a Wavelet neural network respectively. Few literatures have applied a probabilistic approach to modelling and controlling AC loads. [21] attempted to determine the uncertainty of an AC system using Kalman Filters in a white-box modelling setting. In [22], a black-box Gaussian process approach is used to forecast building AC load with uncertainty, before a stochastic model predictive control is used to optimized building AC setpoint.

Note that AC setpoint optimization in a model predictive control (MPC) framework often involves solving an optimization problem at each time step to choose the best AC setpoint over a certain time horizon, with regards to constraints put out by the system dynamics models as well as limitations due to human comfort and electricity prices [23, 24]. Since AC dynamic models are often complex and non-linear, the resulting optimization becomes non-convex and computationally expensive. In [25], an MPC based on the EnergyPlus building simulator is developed, which is limited to a look ahead horizon of one-time step for real-time operation. Otherwise, it will take an entire day to plan the optimal control strategy for the next day. Now, instead of solving for the optimal action at each step, reinforcement learning can serve as an alternative to MPC for decision optimization. Reinforcement learning has been applied in recent years to cloud computing [26], UAV protocol transmission [27], and other smart cities services [28]. When reinforcement learning reaches a certain state, it explores the whole state-action space before deciding on an optimal policy to take a certain action. This policy is typically approximated by a neural network and is suitable for real-time implementation due to its flexibility to be precomputed on a powerful computer, shrunk down, and implemented on a smaller and less powerful discrete controller. Further, in [29], Q-learning is used as an optimization algorithm to control building AC setpoint and room ventilation with models supplemented by EnergyPlus simulation. [30] uses a customized Q-learning algorithm to automate control of heat-pump in residential settings. However, the reinforcement learning literature on AC control do not consider the uncertainty factor of the actions that they undertake.

In summary, the following gaps are identified from existing literature that requires further investigation:

1. Current data-driven techniques used in AC forecasting overlook the reliability of the estimate and does not deal well with data imbalances.
2. Optimization of AC setpoint is usually done by MPC which has to be run with a huge computational cost at each time step. At the same time, most literature focus on centralized AC systems for a whole building or singular AC system with on-off compressors.
3. There is a gap in the literature in AC control for split-type AC systems with variable speed (inverter) compressors that are commonly found in residential houses.

Now, to complement the existing studies on AC control, this paper proposes an automating AC control scheme for split-type AC systems by making the following contributions:

- Aggregating data from all households in a neural network training framework before building individual models for individual households that helps to reduce data imbalances and overfitting.

- Using a Bayesian Convolutional Neural Network (BCNN) to model AC and room temperature such that we are able to take into account the random nature of AC components and model the uncertainty for planning.

- Training a Q-learning reinforcement agent for automated AC control that takes into account uncertainty from the generated models.

The rest of the paper is organized as follows. Section. 2 discusses the system model, where the modeling and control framework is introduced. Section. 3 provides the models for power prediction and room temperature forecasting. In Section. 4, we introduce the proposed reinforcement learning framework. Section. 5 presents the results of the proposed approach in a case study, and finally the study is concluded in Section. 6.

1. SYSTEM MODEL

1.1. Data Description

The dataset that is investigated in this paper consists of 10 single-unit load power readings (P) from 10 residential units in the Singapore University of Technology and Design [5]. These units have the same floor area and uses the same type of inverter split-type AC, the Panasonic CU-S24PKZ. Data from each unit are read off propriety Panasonic IOT sensors, and consist of the following features: AC setpoint (SP), AC power status (PS), indoor room temperature (IT), outdoor temperature (OT), outdoor humidity (H), the number of 30secs blocks since the AC has been powered on ($Tonsin$), and the number of 30secs blocks since the AC has been powered off ($Toff$). The data frequency is every 10 mins, with the dataset being continuous between 1 Apr 2015 to 31 Dec 2016. The ranges of each feature in the dataset is detailed in the Table 1. To facilitate machine learning, the data is min-max normalized to 0-1 according to their ranges. P, SP, PS, IT, $TonSin$, and $Toff$ are data captured off each AC by means of a wireless sensor network. OT and H are weather data supplied by a weather station in Changi, Singapore, scraped off www.wunderground.com.

Table 1: List of features at 10 mins frequency

Features	Min	Max	Units
AC Power (P)	0	3000	Watts
AC SetPoint (SP)	16	30	°C
AC PowerStatus (PS)	0	1	*Boolean*
Indoor Temperature (IT)	16	40	°C
Outdoor Temperature (OT)	16	40	°C
Outdoor Humidity (H)	0	100	%
Time on Since ($TonSin$)	0	240	no. 30s blocks
Time off ($Toff$)	0	240	no. 30s blocks

1.2. The proposed framework

The general idea of this paper is to obtain uncertainty-aware power and temperature models for each room, before planning the optimal SP for each room using reinforcement learning. The general modelling framework is shown in Fig. 1. While looking at the AC consumption pattern of a single room, we find that the consumers typically keep to a narrow set of setpoints, as illustrated in Fig. 2. For example, the residents in Room 5 has been using

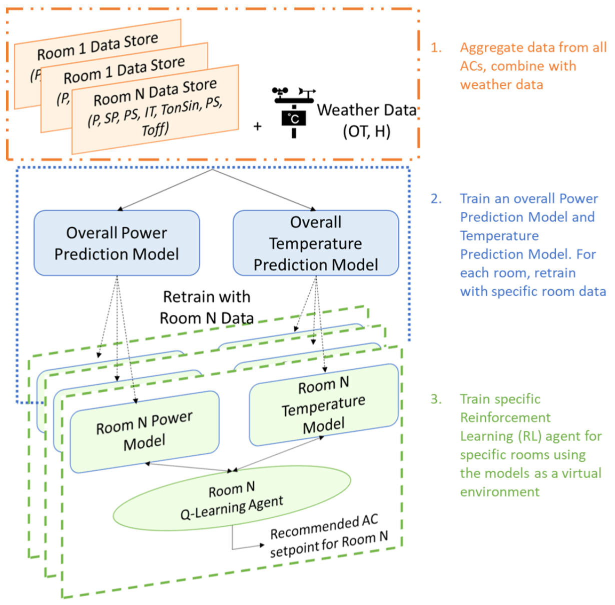

Room 1 Data Store
Room 1 Data Store
Room N Data Store
(P, SP, PS, IT, TonSin, PS, Toff)
+
Weather Data
(OT, H)
1. Aggregate data from all ACs, combine with weather data
Overall Power Prediction Model
Overall Temperature Prediction Model
Retrain with Room N Data
2. Train an overall Power Prediction Model and Temperature Prediction Model. For each room, retrain with specific room data
Room N Power Model
Room N Temperature Model
Room N Q-Learning Agent
Recommended AC setpoint for Room N
3. Train specific Reinforcement Learning (RL) agent for specific rooms using the models as a virtual environment

Figure 1: AC Modelling Framework

their AC at mostly 23 °C for the entirety of the observation period. If we build a black-box model for each room based on the limited and incomplete dataset for each room, we risk a high chance of generating erroneous predictions P as the model might not have an idea on the AC behavior for other setpoints outside of the data it has seen. The first step in our methodology is to aggregate the data from all residences and combined the AC data with weather data. This increases the chances of us obtaining a more complete model for room power and temperature prediction, since each consumer is

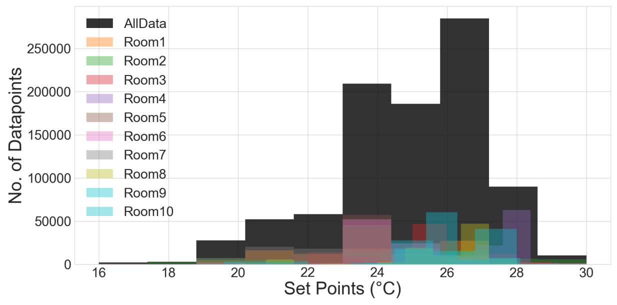


Figure 2: Setpoints distribution across all rooms

using slightly different preference for their AC setpoint.

In the second step, we train an overall Room Power and Room Temperature model based on the aggregated data, before retraining the models with specific room data so that the model can fit more closely to the room dynamics, while retaining information on other setpoints outside the limited dataset of a specific room. The choice of model used is the Bayesian Convolutional Neural Network (BCNN), to be further explained in Section. 3, which allows the neural network to express its uncertainty if the data is scarce or wildly fluctuating. This will result in having in N different Room Power models and N different Room Temperature models for N different rooms.

Finally, with the Room Power and Room Temperature models ready, a Q-learning agent will be developed for each room. Together with historical weather data, the Room Power and Room Temperature model serve as a virtual environment, where the Q-learning agent can sample transitions from. After repeated exploration of this virtual environment, the Q-learning agent will be able to learn and choose the best action at any given state based on a specific reward function. The Q-learning algorithm for this purpose is described in detail in Section. 4. Room 5 and Room 8 were randomly chosen to serve as case studies.

2. ROOM POWER & ROOM TEMPERATURE MODELLING

2.1. Uncertainty-Aware Neural Networks

Artificial neural networks (ANN) have been a popular choice for systems modelling problems due to it's ability to approximate any non-linear process to a high degree of accuracy [15]. By using ANNs to model AC systems, users do not have to actively know the heat gain of the room, the thermal mass and nor the physical control characteristics of the AC, for which they can be implicitly modelled in the model. This allows for the data-driven calibration of any arbitrary room and AC system. However, one of the main gripes of the traditional ANN is that it only supports point forecasts, making it overly confident with regards to unseen scenarios. One way to enable a neural network to express its uncertainty over its input data is to replace all the deterministic weights of a network with a distribution over its weights, and doing back propagation with a process known as variational inference [31]. However, compared to the regular gradient descent for normal neural networks, variational inference is prohibitively slow. In [32], Gal et al. proved mathematically that by applying dropout to fully connected neural networks during training time and testing time, and doing Monte Carlo sampling to the resulting network, we can approximate Bayesian neural nets (BNN) and expresses uncertainty when there is little or conflicting data. They also found that the probability of dropout for BNNs does not matter as the uncertainty estimates will converge in the end. To reduce the computational overheads with regards to Monte Carlo sampling, [33] proposed and proved the mathematical viability of using of a dual output neural network to estimate the bayesian dropout neural network proposed by [32], with one output estimating the mean of the neural network and the other estimating the variance.

In this paper, we leverage the idea proposed by [33]. Considering a normal feedforward neural network Y with D hidden layers, with W^{k}_{k-1} weight matrix linking layer $k-1$ to layer k, $k \in \{1, ..., D, D+1\}$, where layer 0 being the input layer and layer $D+1$ being the output layer. We approximate a Bayesian Neural Network Y^* by applying a dropout matrix $diag(d^k)$ to each weight matrix as per Eq. 1. W^{k*}_{k-1} is scaled by by $1/(1-p)$ so that the output of the weight layer will maintain it's expected scaling, as without dropout. Each element of the diagonal matrix $diag(d^k)$ is sampled from a Bernoulli distribution with dropping probability p, where $0 < p < 1$ in Eq. 2. If the result of the Bernoulli sample is 1, the weight within the weight matrix is kept; and when it is 0, the weight within the weight matrix is dropped. Adding dropout to a fully connected neural network is akin to the creation of an ensemble of neural networks with different weight connection matrices, with each neural network in the ensemble predicting slightly different values due to having dropped weights connection. The output neurons will be responsible for predicting the mean and variance of this ensemble of network.

$$W^{k*}_{k-1} = diag(d^k) * W^{k}_{k-1} \cdot 1/(1-p) \tag{1}$$

$$B(j;p) = \left\{ \begin{array}{cl} p & \text{if } j = 0 \\ 1-p & \text{if } j = 1 \end{array} \right. \tag{2}$$

datapoints refers to the number of samples considered at each batch. At layer $N+1$, the output neuron Y_{mean} for estimating mean value has the standard mean squared error (MSE) loss function, with a linear activation function:

$$\sum^{datapoints} (Y_{actual} - Y_{mean})^2 \tag{3}$$

, and the output neuron Y_{var} for estimating variance minimizes the negative log likelihood (NLL):

$$\sum^{datapoints} (log(Y_{var}) + \frac{(Y_{actual} - Y_{mean})^2}{Y_{var}}) \tag{4}$$

In order to enforce positivity, Y_{var} has the SoftPlus activation function.

Most neural network modelling problems select features from a set of engineered features to improve modelling accuracy. However, the process to engineer and select from such features is tedious. To solve this problem, we designed a double convolutional layer, which can help to automatically extract a higher level set of features that improves model performance. The first convolutional filter has a kernel size of 6 to look for long duration features that occur throughout the hour, while the second convolution filter has a kernel size of 4 to look for shorter duration features that occur within 40 minutes. The output of the convolutional layers is in the form of a 2D array, while the input to the dense layers with dropout only takes in 1D arrays. In order to combine the convolutional layers with the dense layers with dropout, we flatten the 2D output of the convolutional layers into a 1D array before applying dropout on it. With x^{k-1} being the the input and x^k being the output of the k layer respectively, a convolutional layer is represented by Eq. 4 and a dense layer with dropout is represented by Eq. 5. $Total\ Filters$ refers to the number of filters considered for that convolutional layer.

$$x^k = a\,(\sum^{Total\ Filters} g^k * x^{k-1} + b^k\,) \tag{5}$$

b^k represents the bias vector of each layer, g^k represents the 1D convolutional kernel for layer k, $*$ represents the convolutional operator, and $a(.)$ represents the activation function. The rectified linear unit (ReLu) activation function is used as the activation function in this work. $Total\ Units$ refer to the number of filters considered for that dense layer.

$$x^k = a\,(\sum^{Total\ Units} W^{k*}_{k-1} x^{k-1} + b^k\,) \tag{6}$$

The neural network architecture used for both room power modelling and room temperature modelling is shown in Fig. 3. The performance of the neural network is evaluated using the metric known as Mean Absolute Percentage Error (MAPE) as defined in Eq 7.

$$MAPE = \frac{100\%}{datapoints} \sum^{datapoints} |\frac{actual - predicted}{actual}| \tag{7}$$

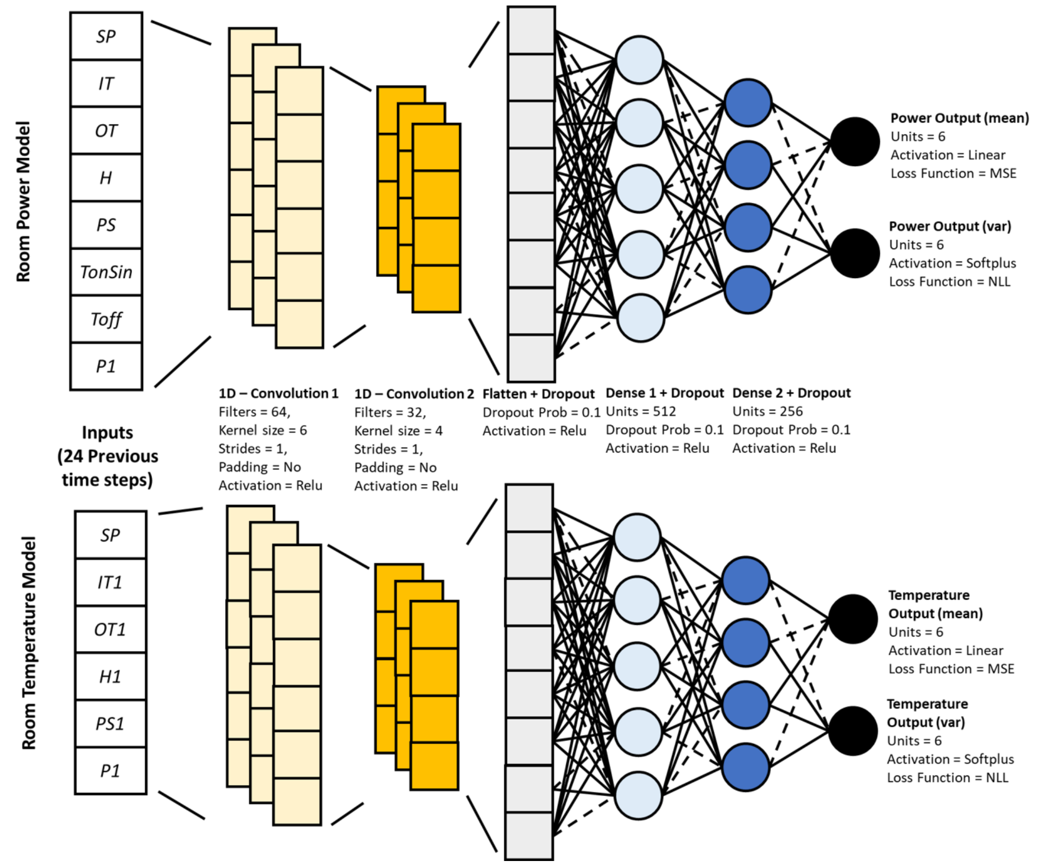


Figure 3: Bayesian Convolutional Neural Network Architecture

2.2. Room Power Modelling

Since the AC for all the rooms are of the same type, and we have selected rooms of the same size, it is highly likely the data for different rooms and setpoints are based on the same physics model of the AC compressor and room structure. Therefore, we attempt to combined all the data from different rooms to estimate this physics model (Combined Model), before fine tuning the model to suit the dynamics of each room (Retrained Model). The Room Power Model aims to predict AC power consumption P of each room for the current time slot based on the input features, SP, IT, OT, H, PS, $TonSin$, $Toff$ and $P1$ (P of the previous time slot). Before we begin training the model, we separate the dataset into 3 different dataset: a training set, a validation set, and a test set. The training set consists of data from all 10 rooms from 1 Apr 2015 to 31 Dec 2016 barring the months of Aug 2015, and Mar to June 2016. The validation set will consist of data from only the room of interest with the same time period as the training set. The test set will consist of data from the room of interest in the month of Aug 2015, and Mar to June 2016. The Room Power Model is trained according to Algorithm 1. The input to the neural network consists of a sliding window of 24 timesteps of the input features, from the previous 23^{rd} timestep back to the current timestep. Structure of the neural network is shown in Fig. 3. By training the neural network on a n-step look ahead output, the chances of the neural network mapping a naive output to the previous power input decreases, and the neural network is less prone to output a time-lagged version of itself [34]. Output of the neural network predicts 6 outputs, the power prediction of the current time step and also the power predictions for the subsequent time steps, but only the first output is used for evaluation.

A regularization technique used for training the neural networks is early stopping. This refers to stop the training of neural network if the loss on the validation set does not change or actually increases over a window of training epochs [35], ie. 10 epoches were used in this paper. The loss considered for early stopping is a summation of the MSE loss in Eq. 3 and the Negative Log Likelihood in Eq. 4. The purpose of first training the room power neural network model on all the data is to allow the model to get a big picture view of all the possible setpoints, and the retraining on the specific room training set allows the network to settle on dynamics specific to the room prediction. The various networks throughout the training process are evaluated and presented in Table 2.

For comparison, we evaluated a version of the bayesian neural network without the CNN layer (BNN), with the input layer being directly connected to the flatten layer, and also with a BCNN neural network trained directly on the validation set (specific room data) without the training set (all room's data). As we can see from the results in Table 2, retraining of the neural network with a dataset more targeted to the room after training with an aggregated dataset improves result as compared to a neural network trained on the specific room dataset, preventing overfitting to the specific dataset due to class imbalances. This is seen in Table 2, where the Retrained Model have lower error than that of the Combined Model across various datasets. Also, addition of a convolutional layer helps in increasing modelling accuracy for both rooms, by around 10%, as seen from the error of the Retrained Models using BCNN being less than that of the error of the Retrained Models using

BNN in Table 2. A plot of the residuals between the prediction via the Retrained Model and the actual power consumption in the first week of Aug 2015 is shown in Fig. 4. The power model is fairly expressive with regards to uncertainty, stating a low uncertainty when the AC is switched off and having zeros power consumption, and outputting high uncertainty estimate when the AC is is switched on. Also, most of the error lies within 95% uncertainty margin, represented by the blue line for prediction residuals lying between the maroon lines for uncertainty bounds in Fig. 4.

Algorithm 1 Power and Temperature Model Training

```
1: Initialization: Prepare training data, test data and validation data,
   initialize neural network;
2: Combined Model: Train neural network on training data with early
   stopping based on validation set, evaluated against test set;
3: Retrained Model: Retrain Combined Model on validation set, with
   early stopping based on test set, evaluated against test set;
4: Return: Retrained Model;
```

Table 2: Room Power Modelling Results in MAPE

	Combined Model	**Retrained Model**
Room 5 - BCNN with all data	13.0668	**10.9441**
Room 5 - BNN with all data	19.8135	18.6805
Room 5 - BCNN with 1 room data	-	16.5431
Room 8 - BCNN with all data	12.3226	**10.3010**
Room 8 - BNN with all data	19.8135	18.6805
Room 8 - BCNN with 1 room data	-	16.5431

2.3. Room Temperature Modelling

Similar to room power modelling, we aggregate all the data for different rooms and setpoints, and train an aggregated model first before fine tuning the room temperature model with data pertaining to each room. The Room Temperature Model forecasts the room temperature *IT* for the next time slot based on the input features of the previous time slot: *SP1*, *IT1*, *OT1*, *H1*, *PS1*, and *P1*. The structure of the neural network used is the same as that of the Room Power Model, with the exception of the inputs. The data

Table 3: Room Temperature Modelling Results in MAPE

	Combined Model	**Retrained Model**
Room 5 - BCNN with all data	1.2376	**1.1552**
Room 5 - BNN with all data	9.5991	8.0021
Room 5 - BCNN with 1 room data	-	7.3835
Room 8 - BCNN with all data	1.1378	**0.9125**
Room 8 - BNN with all data	9.3714	8.0853
Room 8 - BCNN with 1 room data	-	5.6250

is separated into a training, validation and a test set with the same intervals as room power modelling, just that with different set of features. Again, the input to the neural network consist of a sliding window of 24 previous timesteps and the neural network is to predict 6 future outputs of IT. The neural network is trained according to Algorithm 1.

The type of models compared is the same as the room power models, expressed in Table 3. For the temperature model, the BCNN performed the best out of the other variants. It is interesting to note that the adding convolution layers increased the performance of the Room Temperature by around 8%, with the Retrained Models using BCNN having a lower error than that of the Retrained Models using BNN in Table 3. Retraining with specific room data also improves performance, compared to just using the model trained on aggregated data, with the Retrained Models having a lower accuracy than that of the Combined Models. Again, uncertainty values are high when the AC is first switched on or off. Residuals for temperature prediction also largely lies within the 95% uncertainty bound, as seen in Fig. 4.

3. Q-learning FOR AC CONTROL

In light of the room power model and room temperature model being complex and non linear, Q-learning is used to optimize the the control of the AC with regards to a cost function that penalize energy consumption, thermal discomfort, modelling uncertainty, and operation smoothness. This is an educated trial and error process, where a Q-learning agent will explore the environment, calculate the expected reward at each state when taking a particular action, and update this value in a table. Over time, the agent will

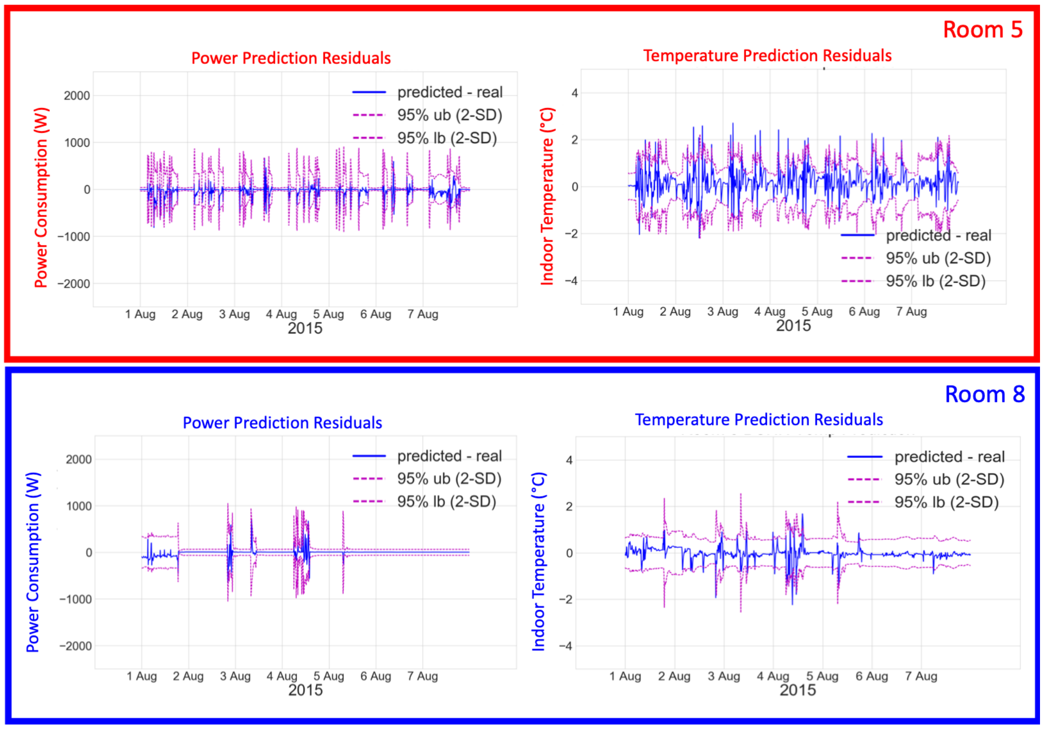


Figure 4: Room Power and Temperature Model Prediction Residuals (Predicted - Actual)

learn which is the best possible action to take in order to obtain the maximum reward over each episode. The variant of Q-learning used in this work uses a neural network to approximate the Q table, incorporate a prioritized experience replay buffer, and double learning as detailed in [36]. Further information on the Q-learning process and values of hyperparameters used in this work can be found in Algorithm 2. Eventually, when the Q-learning agent is deployed in production, real-time performance data can be used to update the BCNNs and at the same time the Q-learning agent, allowing the system to learn autonomously.

3.1. States

We consider each on/off cycle taken by occupants in each room as an episode, and the Q-learning agent is to explore for the optimal SP to be taken at each step in the episode. The states in Q-learning are the observations on the system regarded by the Q-learning agent at each control step. In this study, the states are a 1x22 flatten subset of the 24x8 vector used for

Algorithm 2 Q-learning for Single Room AC Control

```
 1: Initialization: Prepare AC environment for room, initialize neural nets
    $Q(s,a)$, $Q_{aux}(s,a)$, replay buffer $D$ with capacity $D_c$, exploration factor
    $\epsilon$, min exploration factor $\epsilon_m$, decay factor $\lambda$, discounted reward factor $\gamma$,
    number of episodes $E$, and copy steps $C$.
 2: while len($D$) $\leq D_c$ do
 3:    Initialize random state $s_t$, take random action $a_t$, query Room Power
       Model, Room Temperature Model, step through weather data, and
       observe reward $r_t$, store resulting transition ($s_t$, $a_t$, $r_t$, $s_{t+1}$) in $D$
 4: end while
 5: while episodes $\leq E$ do
 6:    Select episode randomly from pool of episodes
 7:    for all step in episode do
 8:       With probability $\epsilon$ select action $a_t$, else $a_t = \arg\max_a Q_{aux}(s,a)$
 9:       Query Room Power Model, Room Temperature Model, step through
          weather data, and observe reward $r_t$ to obtain resulting transition
          ($s_t$, $a_t$, $r_t$, $s_{t+1}$)
10:       Replace transition in $D$ with resulting ($s_t$, $a_t$, $r_t$, $s_{t+1}$) from action
          $a_t$ according to prioritised action replay
11:       Sample minibatch of 64 transitions ($s_j$, $a_j$, $r_j$, $s_{j+1}$) from $D$ based
          on priority
12:       Set $y_j = r_j$ if $s_{j+1}$ is terminal, else $y_j = r_j +$
          $\gamma Q(s_{j+1}, \arg\max_a Q_{aux}(s_{j+1}, a))$
13:       Run gradient descent to minimize Huber Loss between $y_j$ and
          $Q_{aux}(s_j, a_j)$ with learning rate $\alpha$
14:       Decrease exploration probability with $\epsilon = \epsilon_m + (1 - \epsilon_m)e^{-lambda*steps}$
15:    end for
16:    if step % $C$ == 0 then
17:       Copy weights of $Q_{aux}(s,a)$ to $Q(s,a)$
18:    end if
19: end while
20: Return: $Q(s,a)$
```

predicting room power and room temperature, together with future outdoor temperature. The state vector is in the form of:

$$\begin{aligned} state = \; & [SP, IT, OT, H, PS, TonSin, Toff, P1, \\ & SP1, IT1, OT1, H1, PS1, TonSin1, Toff1, P2, \\ & OT_{-1}, OT_{-2}, OT_{-3}, OT_{-4}, OT_{-5}, OT_{-6}] \end{aligned} \tag{8}$$

The variables with the postfix x will be the x time lagged version of the variable (eg. Px being P with x time lag), while the variables with the suffix $_{-x}$ will be the forecasted future information (eg. OT_{-x} being OT x steps ahead of time). The room power and room temperature model consist of the virtual environment, where the effect of an action is fedback to the agent. At each step, depending on the SP taken, the room power P and $P1$ in the state vector will be updated according to the room power model, and the room temperature IT will be updated via the room temperature model. Since we are not optimizing for the on/off of the AC but only for the SP at each on/off cycle, the remainder of the state vector are updated based on historical data.

3.2. Actions

The Q-learning agent is to decide on the SP at each step, from 15 different possible SP, from 16 to 30 °C with an interval of 1 ° C. The quality of each action at each state, the Q-learning network, is modelled by two layered neural networks, $Q(s, a)$ and $Q_{aux}(s, a)$ with 64 neurons of 'ReLu' activation follow by 32 neurons of 'ReLu' activation in the hidden layers. The network will take in the state vector and outputs 15 values, one for each possible action. The structure of the network is shown in Fig.5. At every step, the Q-learning agent will look through all the values from $Q_{aux}(s, a)$ and choose the action with the highest Q-value. The Q-learning network is trained with with backpropagation based on Huber Loss [37] that is clipped at -1 and 1.

3.3. Rewards

Choosing the a good reward function r is extremely important in driving convergence in Q-learning. Our reward function consists of five parts. The first consideration is energy consumption, which is P at each step of an episode. The second consideration is thermal comfort. We consider the original at the start of each episode as the preferred temperature of the occupant, and formulate the comfort penalty as the difference between the $SP_{orignal}$ and IT at each step. The third and fourth consideration is uncertainty in

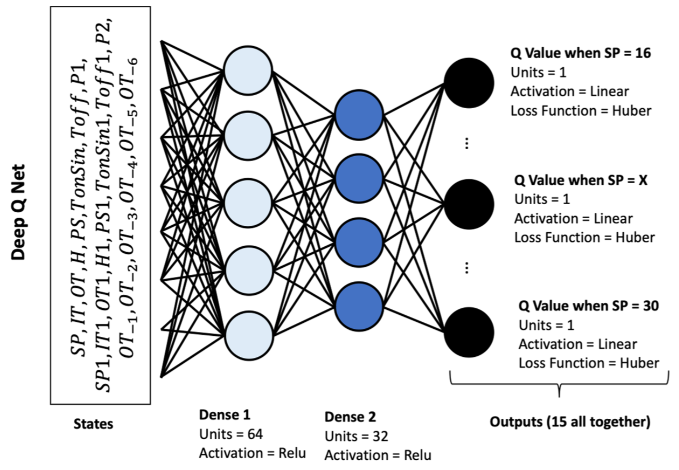


Figure 5: Architecture of DQN for AC Decision

power prediction (P_σ) and temperature prediction (IT_σ). The fifth condition is the smoothness of operation, which penalizes the difference in action for the current time step chosen and the previous time step. Together they are expressed in equation 9 with the constant a, which ranges between 0 to 1 to weigh between saving energy and more thermal comfort. Choosing a higher value for a means more emphasis on power savings.

$$r = \frac{\begin{aligned} & a(1 - P_{mean}) + (1-a)\frac{(3 - abs(SP_{original} - IT_{mean}))}{3} \\ & + a\frac{(0.04 - P_{var})}{0.04} + (1-a)\frac{(0.002 - IT_{var})}{0.002} \\ & + 0.5(max(1 - 3 * abs(SP - SP_1), 0)) \end{aligned}}{2.5} \tag{9}$$

The values of 3, 0.04, 0.02 and 2.5 are normalization factors such that the impact of each factor is weighted and the resulting r ranging between 0 and 1.

3.4. Hyperparameters

Q-learning is an algorithm that learns purely by sampling from the environment, in this case, the developed room temperature and power model. ϵ controls the chance that the Q-learning agent explore the environment. Exploration is generally kept to a maximum at the start of the algorithm to gather information about the dynamics of the system. Once the agent becomes familiar with the system after a certain number of steps, the exploration is decreases exponentially according to the simple ϵ-greedy strategy with constant λ [38] to the minimum exploration rate, ϵ_m. Another constant is γ, ranging from 0-1, which controls how much emphasis the agent takes to future rewards. If γ is kept low, the agent will only select actions that maximise immediate rewards. If γ is high, the agent will try to optimize by taking future actions into account. The learning rate of the agent, α, controls how sensitive the reinforcement agent is to each piece of information, driving convergence to the optimal policy. [39] recommends setting $\alpha = \frac{1}{episodes^{0.8}}$, which is the value used in this paper. Also, the ratio of the number of episodes to run, E, the replay buffer capacity, D_c, and the steps before $Q_{aux}(s,a)$ is updated, C will also affect the learning convergence of the system. The hyperparameters used in the q learning algorithm, Algorithm 2, are summarized in Table 5.

Table 4: Q-learning parameters

Q-Learning Setup						
Offline training (Room Power/Room Temperature Models) to offline control (Q-Learning)						
Q-Learning Training Inputs (April to June 2015)						
Internal States	SP, $SP1$, IT, $IT1$, PS, $PS1$, $TonSin$, $TonSin1$, $Toff$, $Toff1$, $P1$, $P2$					
External States	H, $H1$, OT, $OT1$, OT_{-1}, OT_{-2}, OT_{-3}, OT_{-4}, OT_{-5}, OT_{-6}					
Q-Learning Outputs						
argmax over Q values for each setpoint (16,17,18...30)						
Hyperparmeters						
D_c	ϵ_m	λ	γ	E	α	C
20000	0.01	0.001	0.85	100000	$\frac{1}{episodes^{0.8}}$	100

4. SIMULATED CASE STUDY

The Q-learning agents are trained using historical weather conditions and user on/off sequences in the month of April to June 2015, before being evaluated against the normal user defined setpoint data in the month of Aug 2015 to compare between uncertainty aware and uncertainty unaware Q-learning. A special 50hr sequence in the month of Oct 2015 is also taken for evaluation to compare between uncertainty-aware Q-learning with a rule-based control scheme as detailed in [5]. This 50 hour sequence is a set of real-world experimental results where an energy saving algorithm is actively controlling the user's AC setpoint in the real-world after user has given their permission. We also investigated the effect of considering uncertainty in Q-learning before the effect of changing the a parameter in the reward function, and its performance compared to a rule-based control scheme. The various data used for different evaluations are summarized in Table 5.

Table 5: Comparison Summary

	Agents for comparison	**Training Inputs**	**Testing Inputs**
Comparison between uncertainty-aware Q-learning vs. uncertainty-unaware Q-learning	uncertainty-aware Q-learning agent compared to user on/off sequences in Aug 2015	historical weather conditions and user on-off sequences from April 2015 to June 2015	historical weather conditions and user on-off sequences in the month of Aug 2015
	uncertainty-unaware Q-learning agent compared to user on/off sequences in Aug 2015	historical weather conditions and user on-off sequences from April 2015 to June 2015	historical weather conditions and user on-off sequences in the month of Aug 2015
Effect of adjusting a for power vs. comfort in uncertainty-aware Q-learning	uncertainty-aware Q-learning agent with a = (0.1, 0.3, 0.5, 0.7, 0.9) compared to user on/off sequences in Aug 2015	historical weather conditions and user on-off sequences from April 2015 to June 2015	historical weather conditions and user on-off sequences in the month of Aug 2015
Comparison between uncertainty-aware Q-learning vs. Rule Based Control	uncertainty-aware Q-learning agent compared to rule-based control agent [5]	historical weather conditions and user on-off sequences from April 2015 to June 2015	historical weather conditions and user on-off sequences of 50 hrs in Oct 2015

4.1. Comparison between uncertainty-aware Q-learning vs. uncertainty-unaware Q-learning

For uncertainty-unaware Q-learning, we do not consider uncertainty in our reward function while keeping the rest of the terms, resulting in the following equation:

$$r = \frac{\begin{array}{l} a(1-P_{mean}) + (1-a)\dfrac{(3-abs(SP_{original}-IT_{mean}))}{3} \\ +\,0.5(max(1-3*abs(SP-SP_1),0)) \end{array}}{1.5} \tag{10}$$

A reason for including uncertainty in the reward function is to allow the Q-learning agent to differentiate between the effects of selecting a particular AC . When using a uncertainty-aware reward function, an action leading to high uncertainty in power and room temperature prediction will have less chance to be proposed by the Q-learning agent, leading to a narrower range of operation with smaller power and temperature fluctuations. We are interested to see how this rewards function perform compared to the

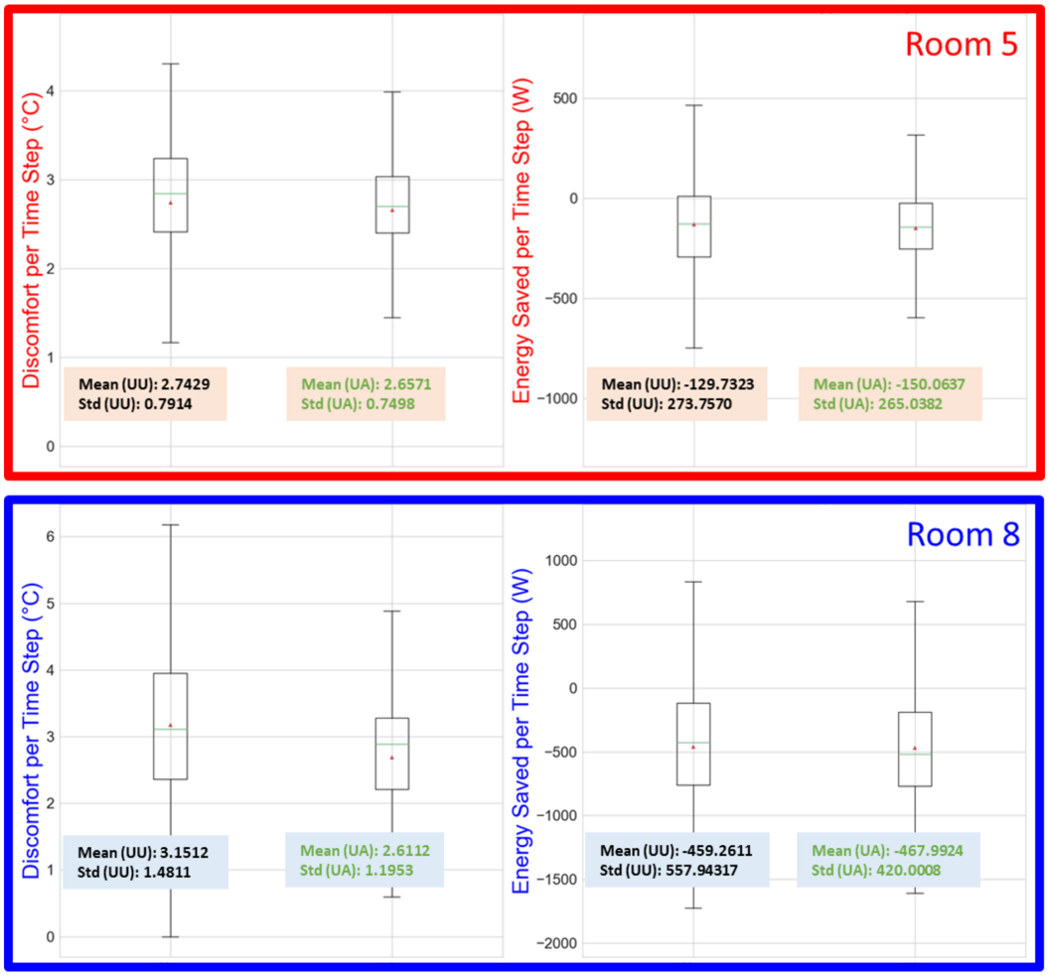


Figure 6: Spread of discomfort and energy saved per time step for Uncertainty-Unaware (UU) and Uncertainty-Aware (UA) agents

uncertainty-unaware reward function. We set the a value to 0.5 to evaluate uncertainty-aware (UA) Q-learning that uses the reward function defined in Eq. 9 and uncertainty-unaware (UU) Q-learning that uses reward function defined in Eq. 10. Boxplots of the control actions taken by the each agent is shown in Fig. 6. Training of the agents were done with on/off sequences in the month of April to June 2015, before being evaluated with user defined control data in the month of Aug 2015.

Discomfort per Time Step refers to the absolute difference between indoor temperature and preferred after taking an action at each time step. Energy Saved per Time Step refers to the difference between energy consumed by the proposed control action by the Q-learning agents and the energy consumed by the original user defined control action in Aug 2015. From the plots, we see that in both Room 5 and Room 8, the standard deviation of the UA Q-learning agents are lower than that of the UU Q-learning agents, validating the fact that UA Q-learning agents will have less fluctuations in operation. However, incidentally, the UA Q-learning agents performed better than the UU Q-learning agents in terms of their mean. This means that in this case, using the UA reward function, we are able to optimize AC with more energy savings and discomfort reduction as compared to the UU reward function, with more stable operation.

4.2. Effect of adjusting for power vs. comfort

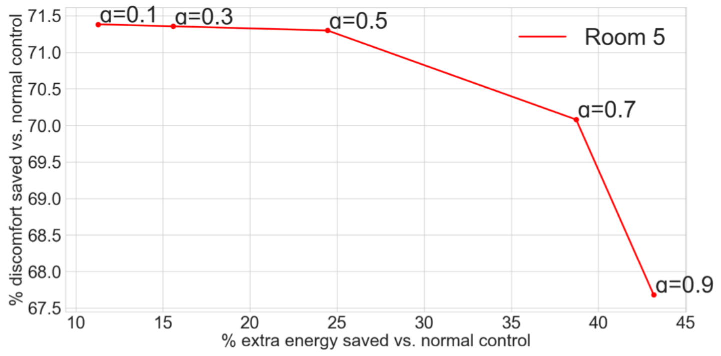
The agent will be evaluated based on the net sum of energy it has saved, as well as the amount of discomfort it has incurred. Similarly, the agents were trained with on/off sequences in the month of April to June 2015, before being evaluated with the user control data in the month of Aug 2015. Averaged discomfort is defined as hourly mean of the absolute difference between indoor temperature and the user setpoint across all timesteps, defined as $\sum_{timesteps=1}^{timesteps=N} abs(IT - SP_{original}) * f$, with f being the number of timesteps per hour. In the month of Aug 2015, user defined control in Room 5 incurred 263.45 kWh of energy, 5.21 °C/hr of averaged discomfort across 342 operating hours, operating at an average setpoint of 23.08 °C. Meanwhile, Room 8 incurred 509.11 kwh of energy, 5.24 °C of averaged discomfort across 388 operating hours, operating at an average setpoint of 21.70 °C. A Q-learning agent was implemented for the operating conditions in Aug 2015 with varying a values of 0.1, 0.3, 0.5, 0.7, and 0.9. Larger values of a place more emphasis on energy conservation over reducing discomfort.

The results for energy saved and averaged discomfort reduced are normalized and presented in Fig. 7. Generally, the Q-learning agents reduced discomfort and energy consumption as compared to using the rule based control in [5]. However, there is a trade-off with regards to optimizing the control actions for energy conservation vs. for comfort. Looking at the gradient of the graph, discomfort levels increase slowly when emphasis on energy conservation is low but increases drastically as more emphasis is placed on energy conservation. In other words, at low values of a ($a \leq 0.5$), users can sacrifice small amount of comfort to achieve good amounts of energy savings. Once past a certain a ($a > 0.5$), users will have to sacrifice a lot more comfort to achieve similar amounts of energy savings. The optimal a for consumers would probably be at 0.5. Different rooms are likely to have the same shape for their energy vs. comfort trade-off curves. However, the actual energy and discomfort conserved in different rooms might be different due to different operating conditions. Factors affecting operating conditions include different setpoints, different time of use, and differences in physical properties of the cooling space due to furniture placement.

4.3. Comparison of UA Q-learning vs. Rule Based Control

A sample control sequence for Room 8 proposed by the Q-learning agent when $a = 0.5$ is compared to the 50hr sequence affeced by the rule based control scheme in [5], shown in Fig. 8. The actual control sequence consumed 20.63 kWh of energy over 50hrs, with 1.56 °C/hr of averaged discomfort. Meanwhile, the control sequence proposed by the Q-learning agent for Room 8 consumed a slightly lower 19.89 kWh of energy, with just 1.44 °C/hr of averaged discomfort. The interesting observation is the Q-learning agent has learnt to take cues from the future weather conditions given to it (6 time steps in advance), lowering the proposed setpoint when outdoor temperature is expected to be high and increasing the proposed setpoint when the outdoor temperature is expected to be low.

The results of changing a for the 50hr control sequence is shown in Table 6. When a is kept low, the discomfort incurred and power consumed by the UA Q-learning agent is comparable to that of the rule based control in [5]. However, when we increase the value of a, we can achieve a much higher energy savings with the UA Q-learning agent as compared to the rule based control, albeit at the sacrifice of thermal comfort.



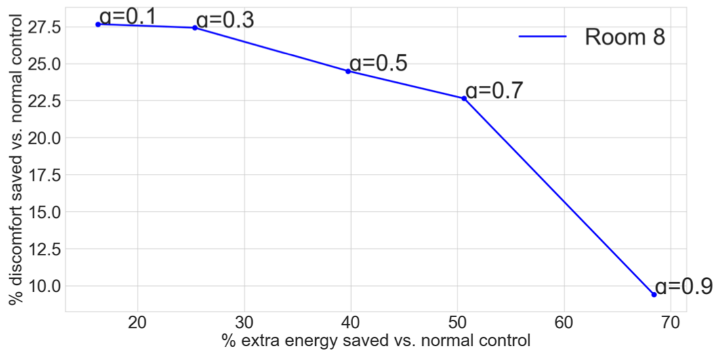


Figure 7: Energy vs. Comfort Trade-off

Table 6: Results of different control scheme applied to 50hr sequence

	Averaged Discomfort per hour (° C)/hr	**Total Power Consumed (kWh)**	**Discomfort improved vs. Baseline (%)**	**Power saved vs. Baseline (%)**
Rule Based Control Scheme in [5] (Baseline)	1.56	20.63	0	0
Uncertainty-Aware Q-Learning, a=0.1	1.50	21.89	3.85	-6.11
Uncertainty-Aware Q-Learning, a=0.5	1.44	19.89	**7.69**	**3.59**
Uncertainty-Aware Q-Learning, a=0.9	3.24	9.69	-107.7	53.0

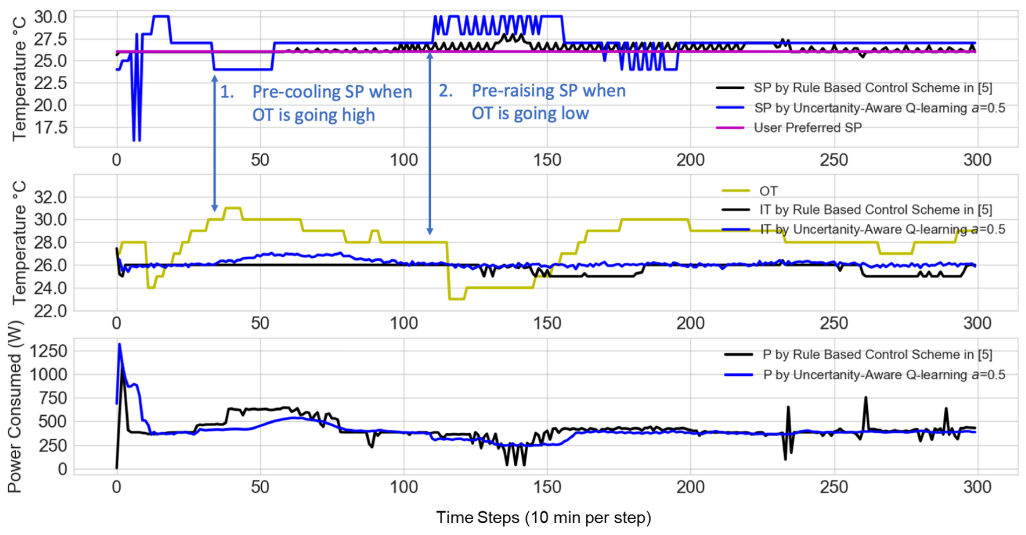


Figure 8: UA Q-learning and Rule Based control for room 8

4.4. Computational requirements

Algorithms described in this paper are written in Python, with the neural networks implemented using Keras and TensorFlow. The computation time for Room Power Model and Room Temperature Model training described in Section. 3 takes around 10 mins running on a Intel i7-7700HQ CPU with a GTX 1060 GPU, for each model. For Q-learning detailed in Section. 4, the training time for each room is around 30 mins. Inference of the Q-learning agent to provide setpoint decision at each time step is relatively fast at 500ms.

The design of the BCNN, and the neural network for the Q-learning agent is important as it will influence the complexity of the algorithm. If the complexity of the BCNN increase by 2 times due to the addition of a few more layers, leading to an slowdown of a factor during BCNN inference, the time taken for Q-learning training will increase by around 2^2 times due to it having to query the network repeatedly during the training process. Meanwhile, complexity of the Q-learning agent neural network is linearly related to the training and inference performance of the Q-learning algorithm.

4.5. Practical limitations of proposed technique

One of the limitations for the generation of the room power and room temperature models in Section 2 is that it requires a dataset with good quality in the distribution of data, more than that of data quantity. Since the purpose of the modelling technique is to solve the data imbalance issue of just observing each room alone, it is better to have data that covers all possible set pints across all outdoor conditions rather than data that only cover a narrow range of setpoints across a limited range of outdoor conditions. In countries where the weather is seasonal, it is important to have at least a year's worth of data. When there is limited information, simulations in building energy software can be performed to supplement the dataset.

Another limitation in the algorithm is that it only suitable for buildings with similar room type and AC types with similar operation modes. When applied to buildings where room types and AC differs, the models in Section 2 and 3 will either have to take on more input variables to differentiate the room types and AC types, or pre-clustering will have to be done to group the data, before having separate models created for each group.

The third limitation is that the Q-learning output layer in Section 3 will have to be modified to match the range of setpoints modelled in Section 2. For example, if the models in Section 2 only models the setpoint of ACs from 15 $^{\circ}C$ to 25 $^{\circ}C$ with 1 $^{\circ}C$ increment, the output layer of the Q-learning network will need to have 10 outputs neurons to represent each increment in the range. If the setpoints of AC between 15 $^{\circ}C$ and 25 $^{\circ}C$ are desired to have 0.5 $^{\circ}C$ increment, the output of the Q-learning network will have to be change to 20 neurons.

In its vanilla form, the algorithms described in this paper are more suited to buildings where room types and AC types are similar. This includes hostels, hotels, or condominiums with homogeneous management-installed cooling systems.

5. CONCLUSION AND FUTURE WORK

In this work, we have achieved our goal of creating an automated data driven AC controller. We have shown that by aggregating data from multiple rooms in a neural network training framework, we are able to reduce the the problems of overfitting and data imbalances. The Bayesian Convolutional Neural Networks (BCNN) used for Room Power and Room Temperature Modelling are able to express uncertainty with regards to different states. Using these models as a simulation environment, the Q-learning agents trained with a uncertainty-aware reward function have shown potential in reducing energy consumption of ACs while preserving human comfort, with flexibility towards energy savings or discomfort reduction by changing the parameter a in the reward function.

A potential limitation of our framework is the large data pool required for successful implementation. However, with the increasing popularity of IoT, as well as more cities in the world subscribing to smart cities initiative, it is easier to collect data. Cities in Europe [40] and Singapore [41] have the ambition to push for zero energy buildings through the installation of smart meters and other related infrastructure, and this could be a potential source of information for our framework. Another advantage of using a neural network based approach is that once new data is available from a real-world setup, we could conduct transfer learning to retrain the model to adapt them to real-world conditions. Our framework is also easily transferable between setup to setup, due to the basic parameters that we use for our AC Power and Temperature models, which are common across different types of ACs. As future work, we would like to extend our framework to another real-world setup for further testing, if the opportunity arises.

Currently, this work is consumer-centric in the sense that it optimizes energy and comfort for the benefit of the consumer. However, the Q-learning agents can be incorporated with dynamic pricing for the ACs to participate in grid level Demand Response [42]. Also, with knowledge of the Room Power and Room Temperature models of each individual residence, the Grid Operator can plan for direct control for ACs or various demand response incentives to benefit the various stakeholders.

ACKNOWLEDGMENT

This work is supported in part by the project funded by National Research Foundation (NRF) via the Green Buildings Innovation Cluster (GBIC), administered by Building and Construction Authority (BCA), and in part by the SUTD-MIT International Design Centre (idc; idc.sutd.edu.sg).

References

[1] Z. Wu, Q.-S. Jia, X. Guan, Optimal control of multiroom hvac system: An event-based approach, IEEE Transactions on Control Systems Technology 24 (2015) 662–669.

[2] P. van den Brom, A. Meijer, H. Visscher, Performance gaps in energy consumption: household groups and building characteristics, Building Research & Information 46 (2018) 54–70.

[3] B. P. L. Lau, S. H. Marakkalage, Y. Zhou, N. U. Hassan, C. Yuen, M. Zhang, U.-X. Tan, A survey of data fusion in smart city applications, Information Fusion 52 (2019) 357–374.

[4] M. Aftab, C. Chen, C.-K. Chau, T. Rahwan, Automatic hvac control with real-time occupancy recognition and simulation-guided model predictive control in low-cost embedded system, Energy and Buildings 154 (2017) 141–156.

[5] W.-T. Li, S. R. Gubba, W. Tushar, C. Yuen, N. U. Hassan, H. V. Poor, K. L. Wood, C.-K. Wen, Data driven electricity management for residential air conditioning systems: An experimental approach, IEEE Transactions on Emerging Topics in Computing (2017).

[6] R. Yin, E. C. Kara, Y. Li, N. DeForest, K. Wang, T. Yong, M. Stadler, Quantifying flexibility of commercial and residential loads for demand response using setpoint changes, Applied Energy 177 (2016) 149–164.

[7] M. Jain, A. Singh, V. Chandan, Non-intrusive estimation and prediction of residential ac energy consumption, in: 2016 IEEE international conference on Pervasive Computing and Communications (PerCom), IEEE, 2016, pp. 1–9.

[8] S. Wang, Z. Ma, Supervisory and optimal control of building hvac systems: A review, HVAC&R Research 14 (2008) 3–32.

[9] Y. Li, Y. Wen, K. Guan, D. Tao, Transforming cooling optimization for green data center via deep reinforcement learning, arXiv preprint arXiv:1709.05077 (2017).

[10] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (1992) 279–292.

[11] W. Zhang, J. Lian, C.-Y. Chang, K. Kalsi, Aggregated modeling and control of air conditioning loads for demand response, IEEE transactions on power systems 28 (2013) 4655–4664.

[12] A. Afram, F. Janabi-Sharifi, Gray-box modeling and validation of residential hvac system for control system design, Applied Energy 137 (2015) 134–150.

[13] C. Lork, Y. Zhou, R. Batchu, C. Yuen, N. M. Pindoriya, An adaptive data driven approach to single unit residential air-conditioning prediction and forecasting using regression trees., in: SMARTGREENS, 2017, pp. 67–76.

[14] Y. Zhou, C. Lork, W.-T. Li, C. Yuen, Y. M. Keow, Benchmarking air-conditioning energy performance of residential rooms based on regression and clustering techniques, Applied Energy 253 (2019) 113548.

[15] A. Afram, F. Janabi-Sharifi, A. S. Fung, K. Raahemifar, Artificial neural network (ann) based model predictive control (mpc) and optimization of hvac systems: A state of the art review and case study of a residential hvac system, Energy and Buildings 141 (2017) 96–113.

[16] C. Cui, X. Zhang, W. Cai, An energy-saving oriented air balancing method for demand controlled ventilation systems with branch and black-box model, Applied Energy 264 (2020) 114734.

[17] M. Fiorentini, J. Wall, Z. Ma, J. H. Braslavsky, P. Cooper, Hybrid model predictive control of a residential hvac system with on-site thermal energy generation and storage, Applied Energy 187 (2017) 465–479.

[18] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, et al., Handling imbalanced datasets: A review, GESTS International Transactions on Computer Science and Engineering 30 (2006) 25–36.

[19] Y. Wang, D. Gan, M. Sun, N. Zhang, Z. Lu, C. Kang, Probabilistic individual load forecasting using pinball loss guided lstm, Applied Energy 235 (2019) 10–20.

[20] M. Rafiei, T. Niknam, J. Aghaei, M. Shafie-Khah, J. P. Catalão, Probabilistic load forecasting using an improved wavelet neural network trained by generalized extreme learning machine, IEEE Transactions on Smart Grid 9 (2018) 6961–6971.

[21] M. Maasoumy, M. Razmara, M. Shahbakhti, A. S. Vincentelli, Handling model uncertainty in model predictive control for energy efficient buildings, Energy and Buildings 77 (2014) 377–392.

[22] A. Jain, T. Nghiem, M. Morari, R. Mangharam, Learning and control using gaussian processes, in: 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), IEEE, 2018, pp. 140–149.

[23] J. Širokỳ, F. Oldewurtel, J. Cigler, S. Prívara, Experimental analysis of model predictive control for an energy efficient building heating system, Applied energy 88 (2011) 3079–3087.

[24] E. Žáčeková, Z. Váňa, J. Cigler, Towards the real-life implementation of mpc for an office building: Identification issues, Applied Energy 135 (2014) 53–62.

[25] Z. Zhang, A. Chong, Y. Pan, C. Zhang, S. Lu, K. P. Lam, A deep reinforcement learning approach to using whole building energy model for hvac optimal control, in: 2018 Building Performance Analysis Conference and SimBuild, 2018.

[26] Y. Sun, M. Peng, S. Mao, Deep reinforcement learning-based mode selection and resource management for green fog radio access networks, IEEE Internet of Things Journal 6 (2018) 1960–1971.

[27] J. Hu, H. Zhang, L. Song, Reinforcement learning for decentralized trajectory design in cellular uav networks with sense-and-send protocol, IEEE Internet of Things Journal (2018).

[28] M. Mohammadi, A. Al-Fuqaha, M. Guizani, J.-S. Oh, Semisupervised deep reinforcement learning in support of iot and smart city services, IEEE Internet of Things Journal 5 (2017) 624–635.

[29] Y. Chen, L. K. Norford, H. W. Samuelson, A. Malkawi, Optimal control of hvac and window systems for natural ventilation through reinforcement learning, Energy and Buildings 169 (2018) 195–205.

[30] F. Ruelens, B. J. Claessens, S. Vandael, B. De Schutter, R. Babuška, R. Belmans, Residential demand response of thermostatically controlled loads using batch reinforcement learning, IEEE Transactions on Smart Grid 8 (2016) 2149–2159.

[31] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight uncertainty in neural networks, arXiv preprint arXiv:1505.05424 (2015).

[32] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: international conference on machine learning, 2016, pp. 1050–1059.

[33] S. Yao, Y. Zhao, H. Shao, A. Zhang, C. Zhang, S. Li, T. Abdelzaher, Rdeepsense: Reliable deep mobile computing models with uncertainty estimations, Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 1 (2018) 173.

[34] K. Amarasinghe, D. L. Marino, M. Manic, Deep neural networks for energy load forecasting, in: 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE), IEEE, 2017, pp. 1483–1488.

[35] R. Caruana, S. Lawrence, C. L. Giles, Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping, in: Advances in neural information processing systems, 2001, pp. 402–408.

[36] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, D. Silver, Rainbow: Combining improvements in deep reinforcement learning, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[37] J. Hwangbo, I. Sa, R. Siegwart, M. Hutter, Control of a quadrotor with reinforcement learning, IEEE Robotics and Automation Letters 2 (2017) 2096–2103.

[38] R. S. Sutton, Generalization in reinforcement learning: Successful examples using sparse coarse coding, in: Advances in neural information processing systems, 1996, pp. 1038–1044.

[39] E. Even-Dar, Y. Mansour, Learning rates for q-learning, Journal of machine learning Research 5 (2003) 1–25.

[40] A. Kylili, P. A. Fokaides, European smart cities: The role of zero energy buildings, Sustainable Cities and Society 15 (2015) 86–95.

[41] A. Bhati, M. Hansen, C. M. Chan, Energy conservation through smart homes in a smart city: A lesson for singapore households, Energy Policy 104 (2017) 230–239.

[42] Q. Wang, C. Zhang, Y. Ding, G. Xydis, J. Wang, J. Østergaard, Review of real-time electricity markets for integrating distributed energy resources and demand response, Applied Energy 138 (2015) 695–706.