A Long-Short Term Memory Recurrent Neural Network Based Reinforcement Learning Controller for Office Heating Ventilation and Air Conditioning Systems

Authors:

Yuan Wang, Kirubakaran Velswamy, Biao Huang

Date Submitted: 2018-07-31

Keywords: reinforcement learning, artificial neural networks, HVAC

Abstract:

Energy optimization in buildings by controlling the Heating Ventilation and Air Conditioning (HVAC) system is being researched extensively. In this paper, a model-free actor-critic Reinforcement Learning (RL) controller is designed using a variant of artificial recurrent neural networks called Long-Short-Term Memory (LSTM) networks. Optimization of thermal comfort alongside energy consumption is the goal in tuning this RL controller. The test platform, our office space, is designed using SketchUp. Using OpenStudio, the HVAC system is installed in the office. The control schemes (ideal thermal comfort, a traditional control and the RL control) are implemented in MATLAB. Using the Building Control Virtual Test Bed (BCVTB), the control of the thermostat schedule during each sample time is implemented for the office in EnergyPlus alongside local weather data. Results from training and validation indicate that the RL controller improves thermal comfort by an average of 15% and energy efficiency by an average of 2.5% as compared to other strategies mentioned.

Record Type: Published Article

Submitted To: LAPSE (Living Archive for Process Systems Engineering)

Citation (overall record, always the latest version):	LAPSE:2018.0252
Citation (this specific file, latest version):	LAPSE:2018.0252-1
Citation (this specific file, this version):	LAPSE:2018.0252-1v1

DOI of Published Version: https://doi.org/10.3390/pr5030046

processes

MDPI

Article

A Long-Short Term Memory Recurrent Neural Network Based Reinforcement Learning Controller for Office Heating Ventilation and Air Conditioning Systems

Yuan Wang, Kirubakaran Velswamy and Biao Huang *

Chemical Engineering Department, University of Alberta, Edmonton, AB T6G 2R3, Canada; yuangary@ualberta.ca (Y.W.); velswamy@ualberta.ca (K.V.)

* Correspondence: biao.huang@ualberta.ca; Tel.: +1-780-492-9016

Received: 13 June 2017; Accepted: 10 August 2017; Published: 18 August 2017

Abstract: Energy optimization in buildings by controlling the Heating Ventilation and Air Conditioning (HVAC) system is being researched extensively. In this paper, a model-free actor-critic Reinforcement Learning (RL) controller is designed using a variant of artificial recurrent neural networks called Long-Short-Term Memory (LSTM) networks. Optimization of thermal comfort alongside energy consumption is the goal in tuning this RL controller. The test platform, our office space, is designed using SketchUp. Using OpenStudio, the HVAC system is installed in the office. The control schemes (ideal thermal comfort, a traditional control and the RL control) are implemented in MATLAB. Using the Building Control Virtual Test Bed (BCVTB), the control of the thermostat schedule during each sample time is implemented for the office in EnergyPlus alongside local weather data. Results from training and validation indicate that the RL controller improves thermal comfort by an average of 15% and energy efficiency by an average of 2.5% as compared to other strategies mentioned.

Keywords: HVAC; reinforcement learning; artificial neural networks

1. Introduction

1.1. Motivation and Background

According to the U.S. Energy Information Administration, the average energy consumed in the buildings sector for residential and commercial users accounts for 20.1% of global energy consumption worldwide. This energy demand is projected to increase by 1.5–2.1% annually between 2014 and 2040 [1]. In the United States, energy consumption from residential and commercial buildings was close to 41% of the total U.S. energy consumption in 2015, or around 39 quadrillion BTU [2]. Of this amount, the buildings' Heating, Ventilation and Air Conditioning (HVAC) systems can account for up to 50% of total building energy demand [2,3]. In the hopes of moving toward a greener, more energy-efficient future, a significant improvement in energy efficiency is needed to achieve this goal. Control of building HVAC systems can lead us one step closer to that goal.

Despite the advances in research on HVAC control algorithms, most field equipment is controlled using classical methods that include hysteresis/on/off and Proportional Integral and Derivative (PID) controllers. These classical methods have been investigated extensively in HVAC lower-loop controls [4–6], such as room temperature control [7–9] and supply air temperature control [10–12]. Despite their popularity, these classical methods do not perform optimally. The high thermal inertia of buildings induces large time delays in the building dynamics, which cannot be handled efficiently by the simple on/off controllers. PID controllers offer an improvement in this respect. However, due

to the high non-linearity in building dynamics coupled with uncertainties such as weather, energy pricing, etc., these PID controllers require extensive re-tuning or auto-tuning capabilities [13], which increases the difficulty and complexity of the control problem.

Due to these challenging aspects of HVAC control, various advanced control methods have been investigated, ranging from gain-scheduling [14], non-linear/robust model predictive control [15–18] to optimal control [19–23]. Between these methods, the quality of control performance relies heavily on accurate process identification and modelling. However, large variations exist with building design, zone layout, long-term dynamics and wide ranging operating conditions. In addition, large disturbance effects from external weather, occupancy schedule changes and varying energy prices make process identification a very challenging problem. This complexity requires one to have expert knowledge in both the underlying systems and controls.

In this work, we introduce novel control algorithms from a branch of machine learning called reinforcement learning. From a controls perspective, reinforcement learning algorithms can be considered as direct adaptive optimal control [24]. Like optimal control, reinforcement learning algorithms minimize the cumulative sum of costs over a time horizon. Unlike traditional optimal control, reinforcement learning algorithms can learn optimal control actions directly through trial-and-error interactions with the plant, without explicit identification of the plant model. By parametrizing the policy and value functions of reinforcement learning algorithms with artificial neural networks, the reinforcement learning controllers can learn to adapt to time-varying and non-linear system dynamics.

In our current approach, the impetus is thermostat control. Instead of traditional on/off heating/cooling control, reinforcement learning is utilized to set this schedule to obtain improved Predicted Mean Vote (PMV)-based thermal comfort at an optimal energy expenditure. Hence, a thermostat schedule is computed using an RL controller.

1.2. Previous Work

The various advantages of reinforcement learning methods have prompted some research directions [25–27] in applying it to HVAC control. For example, Gregor. P et al. [28,29] investigated the use of Q-learning for optimizing thermal energy storage systems in a building. They developed a building simulation platform utilizing MATLAB and EnergyPlus and investigated the use of the tabular Q-learning algorithm. They found encouraging signs of the Q-learning controller learning a desirable load distribution strategy to minimize energy and cost. However, they were concerned with the sample efficiency of the method as compared to a Model-Predictive Controller (MPC). Artificial neural network-based Q-learning was also investigated, but they did not observe improved performance. However, it did improve the sample efficiency compared to the tabular case.

Despite the promising results seen in their work, there were several disadvantages in using the tabular Q-learning method for the HVAC system control problem. Firstly, the Q-learning algorithm requires that the control action space be discretized, and this loses the precise control advantage that a continuous control output algorithm can achieve. Secondly, since tabular Q-learning merely updates Q-value estimates within a finite-sized look-up table, the set of state space representation is limited by the size of the table and is again also discrete. This also reduces the algorithm's ability to generalize to new, unseen state spaces. Thirdly, the Q-learning algorithm implemented assumed state observations to have Markov properties. However, this Markov assumption of full state observation is not applicable in the HVAC control problem, as state observations are both inherently noisy, as well as having high temporal correlations (for example, the indoor zone temperature and outdoor temperature). Lastly, the Q-learning algorithm is deterministic and cannot consider any inherent stochasticity introduced by the external variables, such as weather, occupancy and energy prices.

In our work, we show an improved algorithm and approach that attempt to alleviate all of the issues listed above. Specifically, we introduce two core novelties in our approach:

1. The use of a model-free, actor-critic method that allows for a stochastic policy and continuous controller output. The use of a critic as a variance-reducing baseline also improves upon the sample efficiency.
2. The use of state-of-the-art recurrent neural networks [30,31] for both policy and value representation, which allows for inferring hidden states on noisy observations and incorporating temporal information in state observations

For simulation, we utilize the Building Controls Virtual Test Bed (BCVTB) co-simulation, with the controller implementation in MATLAB and EnergyPlus for building simulation. We simulated an isolated single thermal zone with a simple Variable Air Volume (VAV) HVAC loop, consisting of heating/cooling elements and air blower fans. Low level control is implemented internally within EnergyPlus.

In Section 2, we will provide an overview of the building co-simulation platform developed. In Section 3, we will provide an overview of reinforcement learning, as well as an overview of the algorithms utilized. In Section 4, we will provide an overview of recurrent neural networks and specifically the Long-Short-Term-Memory (LSTM) variant. In Section 5, we will provide a detailed overview of the experimental setup. Lastly, Section 6 present results and discussions on the experimental simulation runs utilizing the platform.

2. Platform Setup

Figure 1 shows a conceptual flow diagram of our building simulation and control platform.

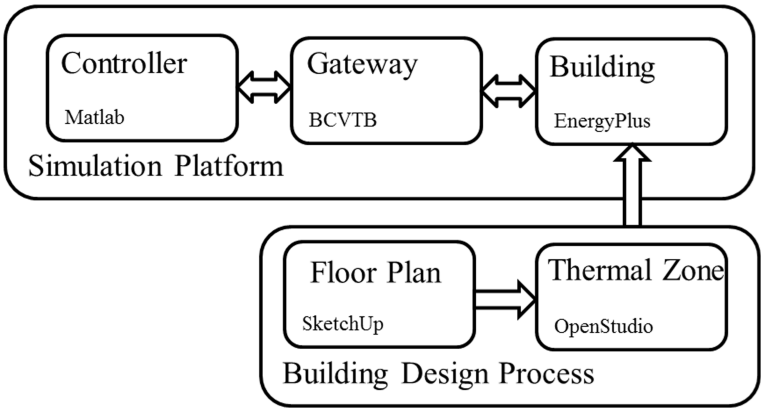

Figure 1. A conceptual flow diagram showing the entire simulation platform design process, including the building design step, as well as the control simulation platform step.

Figure 2 shows the control platform developed. A custom Building Controls Virtual Test Bed (BCVTB) co-simulation allows for communication between building simulation in EnergyPlus and the controller module in MATLAB. An optimization variable module is added to receive raw data from EnergyPlus and computes the optimization cost. In the figure, blue bars are for splitting output vector streams into individual multi-line vector streams, which can then be plotted in the various plotters implemented.

The optimization variable module implemented in MATLAB 2016 uses the Predicted Mean Vote (PMV) [32], which is the thermal comfort measure, and the energy consumed during the sample time to calculate cost. Both PMV and energy values are provided by EnergyPlus to MATLAB via BCVTB.

Additionally, run-time plotters are included in the platform to allow real-time monitoring of the co-simulation ecosystem. The plotters visualize the optimization cost, EnergyPlus outputs and thermostat schedule, as well as controller output. This enabled easier troubleshooting of the platform during initial stages of implementation.

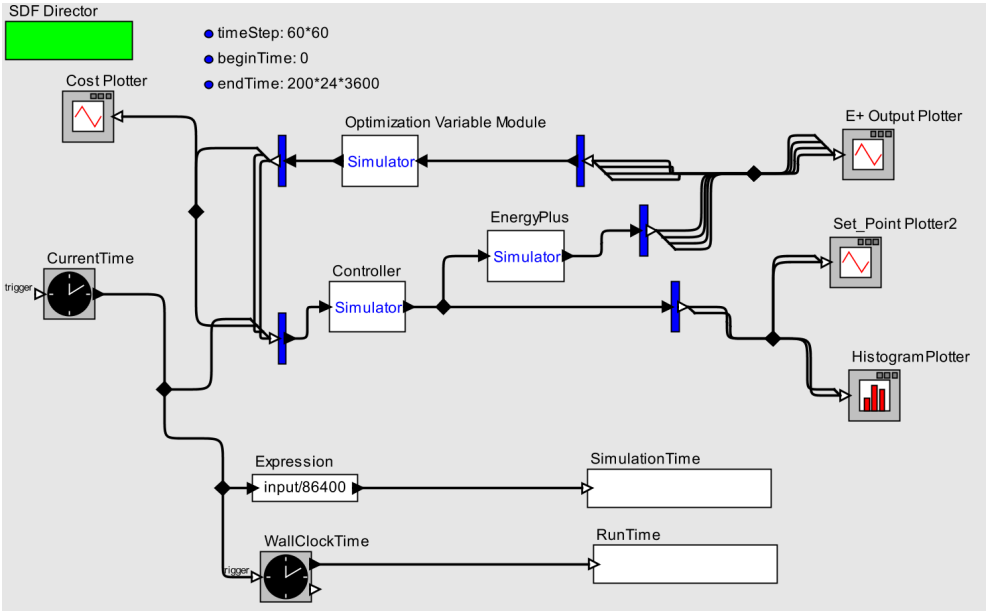

Figure 2. An overview of the BCVTB building simulation setup. It shows three modules (optimization variable, EnergyPlus and controller) alongside various ancillary plotting, time-keeping functionalities. In the figure, blue bars are for splitting output vector streams into multi-line data streams for plotting and data saving uses.

2.1. EnergyPlus

The building simulation is run in EnergyPlus 8.5, using an IDF format files. The process for modelling the building into an IDF starts with the detailed 3D modelling of the building or zone of interest. In this platform, the office space within the Donadeo Innovation Centre for Engineering (ICE) at the University of Alberta has been considered. The thermal zone with attributes representing neighbouring thermal zones is shown in Figure 3.

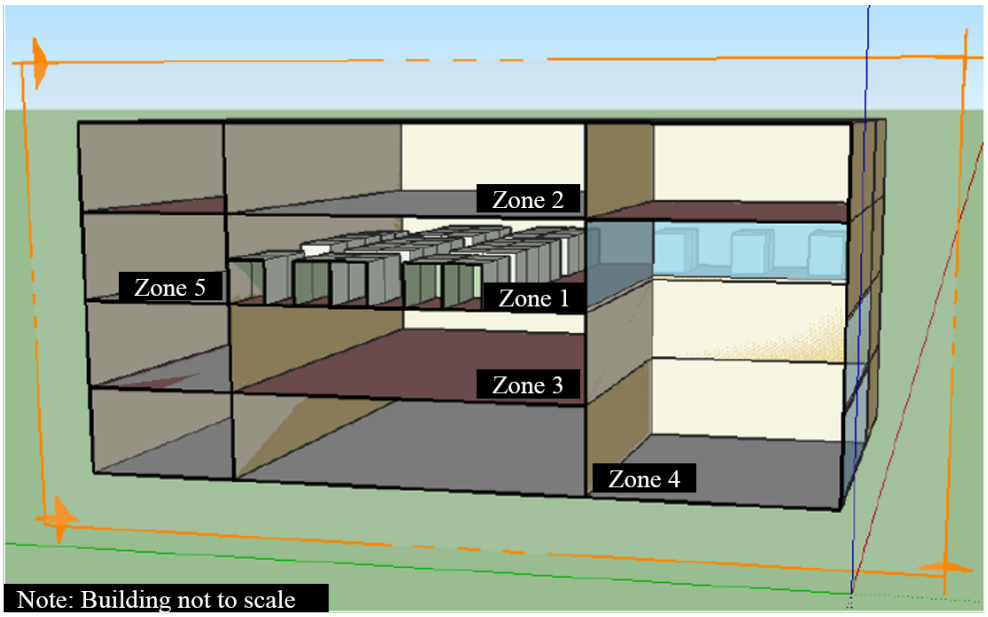

Figure 3. 3D model of our simulated office building zone in SketchUp. Zone 1 is the simulated zone, with Zones 2–5 acting as adjacent, ideal air zones.

During the building design process, using OpenStudio 2.1.0, we also utilized SketchUp 2016 with OpenStudio to specify the space's internal HVAC loop, an isolated single thermal zone with a simple Variable Air Volume (VAV) HVAC loop, consisting of heating/cooling elements, air blower fans and a generic thermostat. See Figure 4.

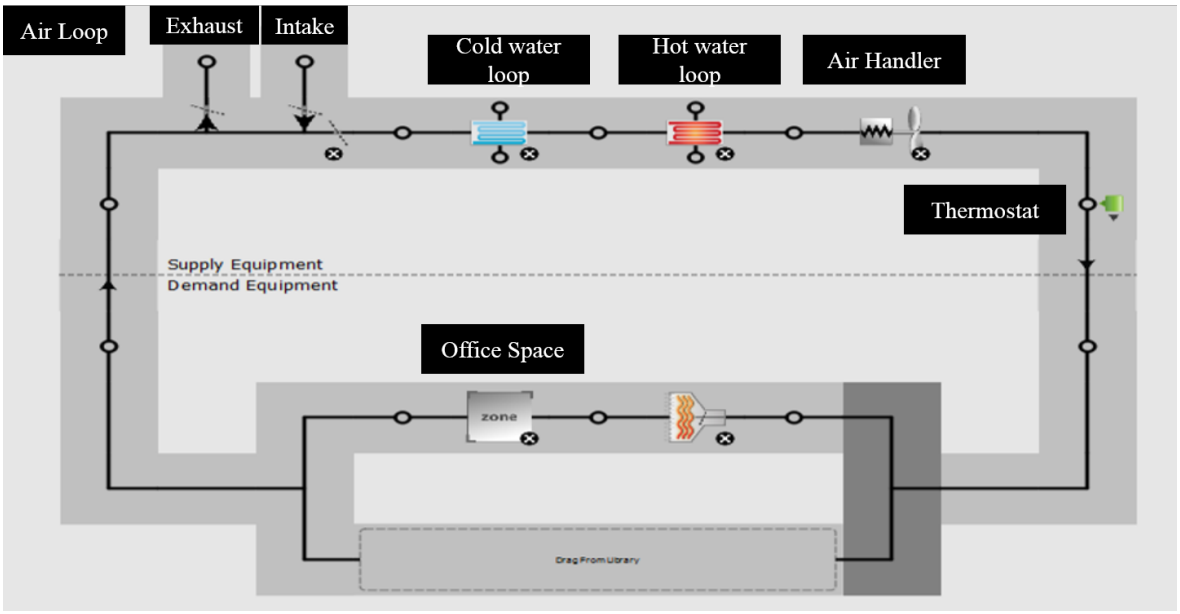

Figure 4. Layout of the simple Variable Air Volume (VAV) HVAC Loop implemented within our room zone simulation. This was a plug-and-use (VAV) module available when designing the building in SketchUp.

2.2. BCVTB

The BCVTB 1.60 co-simulation environment is a Java-based interface to allow for the exchange of data between MATLAB and EnergyPlus. Using the components specifying data flow, plotters, counters and timers, the handshake between EnergyPlus and MATLAB on a sample time basis is implemented. It also allows for initialization and time-out conditions to be specified for the experiment. It allows through the external variable interface in EnergyPlus both higher level (thermostat schedule control) and also access to low level control loops (actuator controls).

3. Deep Reinforcement Learning

3.1. Introduction

Reinforcement Learning (RL) is a computation approach for sequential decision making under uncertainty. Figure 5 shows the interaction between the agent and an environment: at each time step, the agent takes an action and receives a state observation and scalar reward signal from the environment, which is unknown. An RL algorithm tries to maximize the agent's total accumulated reward, given a previously unknown environment, through a trial-and-error learning process.

The description of the general reinforcement learning setup is inherently very general and thus can be applied to a wide range of applications, ranging from robotics [33], game playing [34] to inventory management [35].

3.2. Markov Decision Processes and Value Functions

The Markov Decision Process (MDP) mathematically formalizes the agent-environment interactions. It has the following core components:

- $s \in \mathbb{S}$: state space, the set of all possible states of the environment.
- $a \in \mathbb{A}$: action space, the set of all possible actions, from which the agent selects one at every time step.
- $r \in \mathbb{R}$: reward, a scalar reward emitted by the environment at every time step.
- $\pi(a \mid s)$: policy, which maps from state observation s to action a. Typically, stochastic policies are used; however, deterministic policies can also be specified.
- $\mathbb{P}(s', r|s, a)$: transition probability distribution, which specifies the one-step dynamics of the environment, the probability that the environment will emit reward r and transition to subsequent state s' from being in state s and having taken action a.

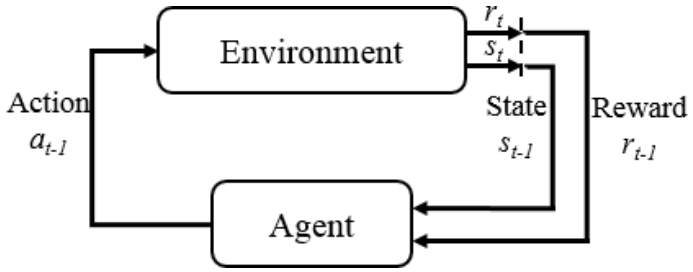

Figure 5. A figure showing the various components in a reinforcement learning problem setting. The dashed vertical line indicates a transition in discrete time/sequence. This diagram shows a full transition in states and reward from one time step to the next.

The goal of the RL agent is to learn an optimal policy that maximizes the expected sum of rewards, also called returns, G_t, shown in Equation (1).

$$G_t \doteq r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_\infty \tag{1}$$

In general, it is common to apply a discounting factor to the returns, called the discount rate γ, $0 \leq \gamma \leq 1$. It acts to determine the present value of the rewards. The smaller the γ, the less future rewards matter, and vice versa. In addition, it acts to threshold the returns to a finite, bounded value. With discounting, the total returns are shown in Equation (2).

$$G_t \doteq r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2}$$

Value functions are important parts of many RL algorithms. In essence, they estimate the expected total returns starting from some state. Two core value functions are present in RL, the first one being the state-value function, $v^\pi(s)$. This function Equation (3) estimates the expected total returns G_t, starting from a certain state s, and following policy π.

$$.v^\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s\right] \tag{3}$$

The other value function of interest is called the action-value function, $q^\pi(s, a)$ Equation (4). Similar to the state-value function above, the action-value function estimates the expected total returns G_t starting from some state s and taking an action a following some policy π.

$$q^\pi(s, a) \doteq \mathbb{E}_\pi\left[G_t|s, a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s, a\right] \tag{4}$$

Many popular RL algorithms rely heavily on the above value functions, which are estimated from experience. Their usefulness lies in that once the optimal value functions, v_π^* or q_π^*, have been obtained from sufficient experience, the optimal policy, π^*, can be found by repeatedly applying the value function for every state and taking the greedy action that leads to the state with the highest value estimate. One example of such an algorithm is Q-learning, which utilizes a parameterized action-value function $q_\pi(s, a)$ to learn the optimal action-value function.

3.3. Policy Gradient

One class of reinforcement learning methods deals exclusively within the parameterized policy space. Specifically, a differentiable parameterized model, such as a neural network model with parameters $\theta \in \mathbb{R}^d$, is used to represent a stochastic policy. This direct policy parameterization allows us to convert the RL problem into that of an optimization problem; namely, maximizing the objective function $J(\theta)$, the total returns, under the policy π_θ given in Equation (5).

$$\max_\theta J(\theta) = \mathbb{E}(G_t|\pi_\theta) \tag{5}$$

The policy gradient method offers several advantages over value-based methods, improving in areas such as stability, direct optimization, etc. There exist multiple ways to estimate the policy gradient, from finite difference methods [36] to likelihood ratio methods [37]. We focus on the latter method, specifically the REINFORCE policy gradient [38]. The likelihood ratio method's advantages are faster convergence rates, and this nullified the need for generating random perturbations in policy parameters, which can lead to instability.

The REINFORCE [38] policy gradient provides an unbiased estimate of the gradient of the policy with respect to the objective function $J(\theta)$ given in Equation (6).

$$\bigtriangledown_\theta J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \bigtriangledown_\theta \log \pi(a_t|s_t, \theta) \sum_{t'=t}^{T-1} r_{t'} \right] \tag{6}$$

One point with the REINFORCE policy gradient is that it has high variance, as it is using the actual returns $r_{t'}$ as a component of the gradient estimate. It can be shown that subtracting the returns $r_{t'}$ by an arbitrary baseline term, $b(s_t) \in \mathbb{R}$, can significantly reduce the variance in gradient estimates [39]. It can also be shown that the baseline introduces no bias in the gradient estimate [39], i.e., $\mathbb{E}_\pi[\bigtriangledown_\theta \log(a_{t'}|s_{t'}, \theta) b(s_t)] = 0$. Thus, by choosing our baseline carefully, one can significantly reduce the variance and, thus, the sample complexity in our gradient estimates. Thus, the general policy gradient estimate with an arbitrary baseline function is provided in Equation (7).

$$\bigtriangledown_\theta J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \bigtriangledown_\theta \log \pi(a_t|s_t, \theta) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right) \right] \tag{7}$$

3.4. Actor-Critic Methods

Actor-critic methods are based mainly on the policy gradient with variance-reducing baseline derivation obtained in Equation (7). A more general representation is to call the term on the right-hand side of Equation (7) the advantage estimator, A_t. The various different actor-critic methods thus differ in how they approximate the advantage estimator. Thus, in actor-critic methods, the actor term refers to a parameterized policy, and the critic term refers to some parameterized advantage estimate.

In our work, we introduce a Monte Carlo actor-critic algorithm that utilizes a state-value function, $v^\pi(s)$, as the critic. The policy gradient estimate is shown in Equation (8).

$$\bigtriangledown_\theta J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \bigtriangledown_\theta \log \pi(a_t|s_t, \theta) \left(\sum_{t'=t}^{T-1} r_{t'} - v^\pi(s_t) \right) \right] \tag{8}$$

The algorithm is updated off-line using the collection of experiences obtained on-line with fixed parameters. Due to the inherent noise and high sequential nature of the HVAC thermostat-schedule problem, we opt to utilize recurrent neural networks for parametrizing both the policy and critic. We utilize recurrent neural networks termed Long-Short-Term-Memory networks to address the issue of noise and the partial observability of states. Since LSTMs maintain a hidden state during computation, this hidden state is able to incorporate historical data to infer the true state of the system, from observing a history of noisy, the partially-observed state observation data.

The pseudo-code for our algorithm is given below:

Algorithm 1 Monte Carlo on policy actor-critic.

Require: Initialize policy π with parameters θ_π and value critic v_π with parameters θ_v

for each episode **do**
  Get initial state s
  Initialize storage buffer S, A, R, S'
  for $i = 1, 2, 3...N$ steps **do**
    Sample action with policy: $a \sim \pi_\theta(s)$
    Run action through environment, obtain reward and post state: $r, s' \leftarrow ENV(s, a)$
    Collect and store: $S, A, R, S' \leftarrow s, a, r, s'$
    $s \leftarrow s'$
  end for
  Compute discount returns: $\hat{V} = \sum_{l=0}^{N-1} \gamma^l r_{t+l}$
  Update θ_v to minimize $\sum_{n=1}^{N} \|v_\pi(s_n) - \hat{V}_n\|^2$
  With learning rate α, update policy: $\theta_\pi \leftarrow \theta_\pi + \alpha \nabla_\theta \log \pi(A|S) v_\pi(S)$
end for

4. Recurrent Neural Networks

Artificial neural networks, specifically the recent branch termed as deep learning, have gained an unprecedented revived interest and attention in the past decade from both academia and industry. They have achieved the state-of-the-art in many fields, ranging from computer vision [40], natural language processing [41] to niche applications, such as self-driving cars [42]. For datasets with a sequential nature, such as for natural language processing, Recurrent Neural Networks (RNN) have been applied successfully due to their capacity to model highly non-linear data, increasing the availability of datasets and power computers. Despite traditional recurrent neural network's inability to handle long sequences of data, a new class of network architectures with learnable gates has been shown to effectively alleviate this problem. The most popular of these variant is the long-short-term-memory network architecture.

4.1. Vanilla Recurrent Neural Network

Artificial neural networks are a class of non-linear function approximations. They are composed of nested affine transforms followed by a non-linear activation transform. Given an input $\boldsymbol{x}$ and layer input weight matrix $\boldsymbol{W_{in}}$, we first apply an affine transform and then compute $g(x)$, a non-linear activation transform, over the affine transform output. The computation is shown in Equation (9).

$$f(\boldsymbol{x}; \boldsymbol{w}, b) = g(\boldsymbol{W_{in}}\boldsymbol{x} + \boldsymbol{b}) \tag{9}$$

Since RNN works on sequential data, it has an explicit hidden state, h_t, that is recursively carried forward and updated as each element of the sequential data is passed through the recurrent neural network. To facilitate this, an additional recurrent weight, $\boldsymbol{W_{rec}}$, is need for this computation step.

For a given sequential data from $t = 1$ to $t = T$, the update Equation (10) is applied for a hidden layer recurrent neural network.

$$\boldsymbol{a}_t = \boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{b} \tag{10}$$

$$\boldsymbol{h}_t = g(\boldsymbol{a}_t) \tag{11}$$

Two points of importance are noted from the computation steps above:

- The same layer weights $\boldsymbol{W}$ and $\boldsymbol{U}$ are re-used for each step of computation throughout the sequential data
- An RNN is similar to a deep feedforward neural network when unrolled through time. The depth will be of the length of the sequential data, and for each layer, the weights are the same

4.2. Vanishing and Exploding Gradient Problem

The back-propagation of error signals through the RNN is similar to that of a feed-forward neural network. Starting from the error output, we go backwards and apply the chain rule to compute the gradient of each layer's parameters using gradients coming from the next layer. For RNN, this is called Back-Propagation Through Time (BPTT). Due to the fact that the same layer weights are re-used for each item in the sequential data, when the computation is unrolled through time, the network can be viewed as a very deep feed-forward network in the time dimension, with shared weights at all layers. Figure 6 shows a diagram of an unrolled RNN over three computation steps.

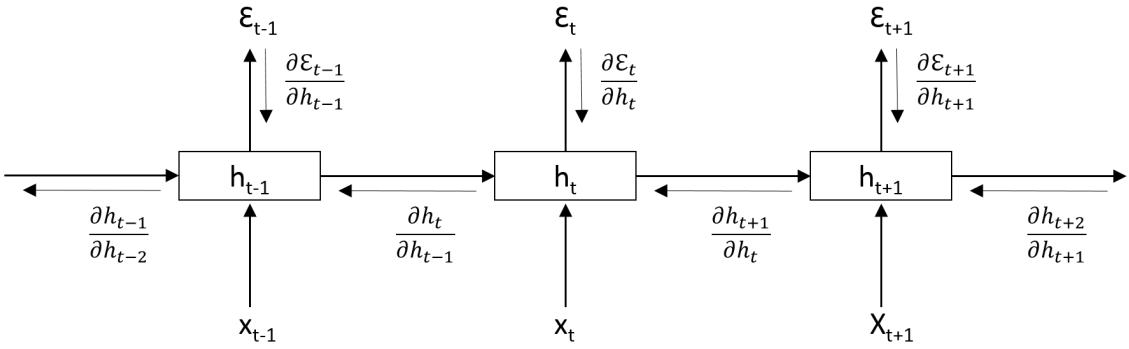

Figure 6. The RNN here has been unrolled for three time steps for some arbitrary sequential data. At every time step, the RNN's output is computed, as well as the hidden state is passed forward through time. During back-propagation, the gradients flow back in two paths; first from error outputs at every time step $\frac{\partial \mathcal{E}_t}{\partial h_t}$, as well as the gradient back-flow through time through the recurrent states, $\frac{\partial h_{t+1}}{\partial h_t}$.

The BPTT algorithm computes the gradient of the total loss, $\mathcal{E}$ across the entire sequence. Since the RNN's parameters are re-used for each element in the input sequential data, the total loss gradient is the sum of the loss gradient for each sequence of time, given in Equation (12).

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \leq t \leq T} \frac{\partial \mathcal{E}_t}{\partial \theta} \tag{12}$$

We can further expand the right-hand side to include two gradient components, first the gradient from the loss at each time step, as well as the recurrent gradient flow back, provided in Equation (13).

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \leq t \leq T} \left(\frac{\partial \mathcal{E}_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial \theta} \right) \tag{13}$$

In particular, when we focus on the temporal gradient flow-back component, $\frac{\partial h_t}{\partial h_k}$, we can see that it is a product of the same gradient starting from time t and extending until the end of the sequence k, as shown in Equation (14).

$$\frac{\partial h_t}{h_k} = \prod_{t \geq i > k} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{t \geq i > k} \boldsymbol{u}^T diag(g'(h_{i-1})) \tag{14}$$

4.3. Long-Short-Term-Memory Recurrent Neural Network

One popular method to alleviate the above-mentioned vanishing gradient problem is LSTM [30,31] networks. LSTM is a special variant of recurrent neural networks that improves upon the traditional RNN in the following ways:

- Learnable gates that modulate the flow of information.
- A persistent cell state that has minimal interactions, providing an easy path for gradient flow during back-propagation.

A total of three gates is utilized within an LSTM; these are the input gate i_t, Equation (16), forget gate f_t, Equation (15) and output gate, o_t Equation (17). To allow for the gates to modulate the flow of information, as well as be differentiable, sigmoid/logistic function σ is used to threshold the gate outputs to between [0,1].

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{15}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{16}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{17}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{18}$$

$$h_t = o_t \odot \sigma(c_t) \tag{19}$$

Once the gates are computed, the cell state c_t is updated by Equation (18) in two parts. The first component is a dot product between the previous cell state c_{t-1} and the forget gate f_t. This modulates how much of the cell state information from the previous time step should be kept. The second part is a dot product between the current cell state computation and the input gate i_t. This modulates how much of the current information should be allowed into c_t. Once the cell state has been updated, the hidden output h_t is modulated by the output gate o_t via another dot product Equation (19). Figure 7 provides a visualization of the LSTM equations.

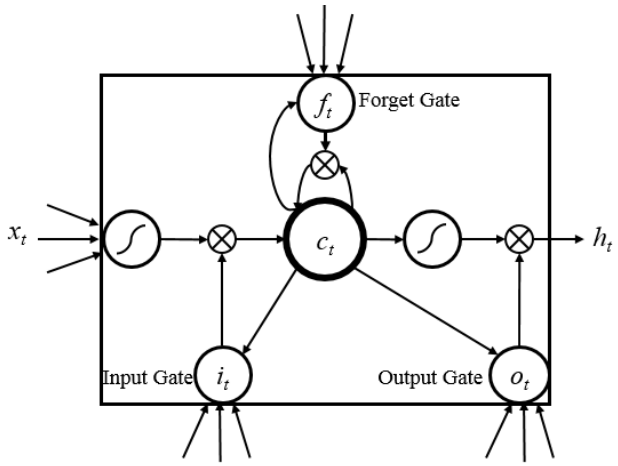

Figure 7. A visual diagram of the LSTM architecture, showing the input, forget and output gates, as well as the cell state computation and updates.

5. Setup

For comparative study, we implemented an actor-critic-based RL controller applied to the building HVAC control and compared its performance against three baselines.

5.1. Simulation Setup and Parameters

We utilized our simulation platform setup for the experimental case study. The simulation was conducted over a period of seven days, two days for training the RL controller and five days for validation testing. We utilized our simulation platform setup for the experimental case study. The simulation was conducted over a period of seven days, two days for training the RL controller and five days for validation testing. We selected this simulation window to ensure no large variations (example: $\pm$10% of 26.53 °C for ambient temperature) in external variables, such as ambient temperature, cloud cover and solar irradiance, between the training set and validation set. A single room zone (Zone 1) with window access is investigated in our simulation. This single zone has thermal contributions from surrounding ideal air zones (Zones 2–5). The simulation sampling time is 5 min. The occupancy considers three people in an office setting, measured as a percentage. Table 1 provides detailed information.

Table 1. This table provides the daily occupancy schedule for a typical work day on an hourly-basis. Values reported in fractions. This occupancy schedule was used in our simulation.

Hours	Office Occupancy	Equipment Active Schedule
01:00 a.m.	0	0
02:00 a.m.	0	0
03:00 a.m.	0	0
04:00 a.m.	0	0
05:00 a.m.	0	0
06:00 a.m.	0	0
07:00 a.m.	0.5	1
08:00 a.m.	1.0	1
09:00 a.m.	1.0	1
10:00 a.m.	1.0	1
11:00 a.m.	1.0	1
12:00 p.m.	0.5	1
01:00 p.m.	1.0	1
02:00 p.m.	1.0	1
03:00 p.m.	1.0	1
04:00 p.m.	1.0	1
05:00 p.m.	0.5	1
06:00 p.m.	0.1	1
07:00 p.m.	0	0
08:00 p.m.	0	0
09:00 p.m.	0	0
10:00 p.m.	0	0
11:00 p.m.	0	0
12:00 a.m.	0	0

For training of the reinforcement learning controller, we utilized two days of simulation data to train the policy and critic LSTM networks in our RL controller. After the training, we tested the performance of the controller against benchmarks in the five-day simulation.

5.2. Reinforcement Learning Controller Design

The reinforcement learning controller is composed of two LSTM networks, one for the actor and one for the critic.

The state observations, which are provided at every control time step to the RL agent, are chosen to be both informative and easy to access. Specifically, the state observations are composed of the outdoor temperature T_a (°C), indoor temperature T_r (°C), the energy demand from the last time step Q_{cool} (W/h) and solar irradiance Q_{sol} (W/m^2). The controller outputs over a discrete control action set of size 26, from 20 °C to 25 °C with 0.2 °C resolution. This control output is then interpreted as the room cooling temperature set point T_c, which will be passed onto the simulation platform.

Figure 8 shows the RL controller architecture setup. We see that the discrete control action output is computed by the actor, given the state observation. The critic, a state-value function v^{π}, computes a single scalar value representing the state-value of a particular state. This scalar value is used as the baseline in the critic's policy gradient update.

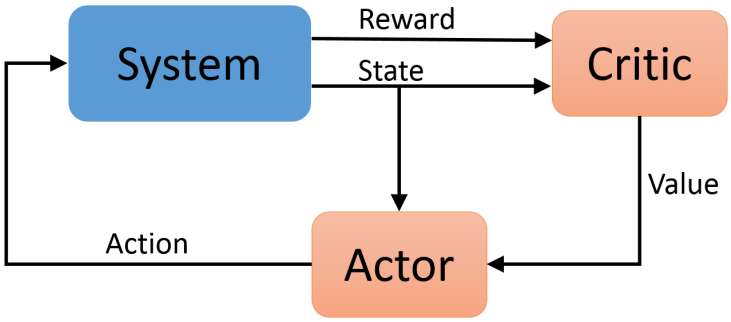

Figure 8. A high-level block diagram of the actor-critic reinforcement learning architecture is shown. This shows the general flow of state observations and reward signals between the algorithm and the environment, the critic's update and its value estimate, which is used by the policy in it's policy gradient updates.

5.3. Optimization Cost Function Structure

The optimization cost function (also referred to as the reward function) design is of great importance in the reinforcement learning problem setup, as an ill-defined cost function can lead to unexpected and undesirable controller performance.

In our study, we would like the controller to achieve thermal comfort while maintaining a certain level of energy efficiency. It is hoped that by relaxing the thermal comfort threshold (τ), the RL controller can achieve energy savings while still ensuring a tolerable level of thermal comfort. To obtain a numerical value for thermal comfort, the Predicted Mean Vote (PMV) developed by P.O.Fanger [32] was used. The PMV calculation takes many factors into account, ranging from air temperature, relative humidity, clothing insulation and metabolic rate. The PMV calculation represents the thermal comfort within a scale from hot[+3] to cold[-3]. Usually, thermal comfort values of between ± 0.5 and ± 1.0 are considered acceptable. In our setup, the simulation platform computes the PMV values internally in EnergyPlus.

The total cost, C, which is computed at every control step, is composed of a weighted sum of the two objectives, PMV (thermal comfort) and energy demand, given in Equation (20).

$$C = \sum_{t=0}^{t=T} (\alpha \cdot PMV_{cost} + (1 - \alpha) \cdot Q_{cool}) \tag{20}$$

The α in the equation is a weighting that balances the contribution from the two component costs, PMV cost (PMV_{cost}) in Equation (21) and energy (Q_{cool}). In our experiment, we set α at 0.5. In addition, PMV_{cost} and Q_{cool} are normalized to be between [0,1].

$$PMV_{cost} = \begin{cases} 0 & \text{for} \quad |PMV| \leq \tau \\ \frac{|PMV|}{3} & \text{for} \quad \tau \leq |PMV| \leq 3 \\ |PMV| - \tau & \text{for} \quad 3 < |PMV| \end{cases} \tag{21}$$

The PMV_{cost} is computed to have zero effect on the total cost if it is below the PMV threshold, τ. Note that since PMV values typically range between ± 5, we take the PMV's absolute value when computing its cost contribution.

The energy demand Q_{cool} is obtained raw from the our simulation tool and is also normalized.

6. Results and Discussion

6.1. Baseline Setup

For comprehensive comparison, two different control baselines were setup to evaluate the effectiveness of the reinforcement learning controller. These baselines vary in the various aspects of the controller performance.

The first baseline is called the ideal PMV baseline, B_{ideal}. This baseline maintains the ideal PMV without regard for the energy usage required to maintain this precise PMV requirement. This is given in Equation (22).

$$B_{ideal} = \begin{cases} T_h = 21 \\ T_c = 23 \end{cases} \tag{22}$$

where T_c is the cooling set point (°C) and T_h is the heating set point (°C).

Secondly, a variable control $B_{variable}$ was implemented. This baseline contains some basic temperature-tracking capabilities. Specifically, it tracks the external temperature and sets T_c according to the following equation if it is within 20 and 25 °C. Since the simulation runs are conducted in the summer month of July, room heating is never required as external temperatures range well above 20 °C. Due to this factor, heating set point T_h is set constant at 16 °C for our baseline setup. Note that T_h can be set up in a similar fashion to T_c for simulation through colder months when room heating is needed.

$$B_{variable} = \begin{cases} T_h = 16 & \\ T_c = 20 & \text{for} \quad T_r < 20 \\ T_c = T_r & \text{for} \quad 20 \leq T_r \leq 25 \\ T_c = 25 & \text{for} \quad T_r > 25 \end{cases} \tag{23}$$

where T_r is the room temperature (°C).

6.2. RL Training Phase

The RL controller was trained using two days of simulation data with a sampling time of 5 min. Figure 9 shows the RL controller's trajectories across each iteration of training. It can be observed that initially, the controller's output contains large variation, as clearly observed in the cooling rate Q_{cool}. With each training iteration, the controller outputs stabilize, until they stabilize to the optimal controller output with low overall cost.

Table 2 provides the metrics associated with the tuning exercise. As described in the optimization cost function in Equation (20), the metrics involving both thermal comfort Equation (21) and energy consumption are provided. With a blind start initially, the iterations continue to improve both measures, limited by the inertia of the building profile. On obtaining a negligible gradient in metrics between consecutive iterations, the procedure is halted.

Table 2. RL training results.

Iteration Number	PMV Total (Unitless)	Energy Total (W)	Standard Deviation (Q_{cool} W/h)
1	216.70	16,717	3580
2	113.50	18,079	3480
3	105.60	18,251	3500
4	262.00	15,398	3070
5	132.12	17,698	3450
6	211.30	16,300	3240
7	211.50	16,310	3240
8	211.30	16,298	3228

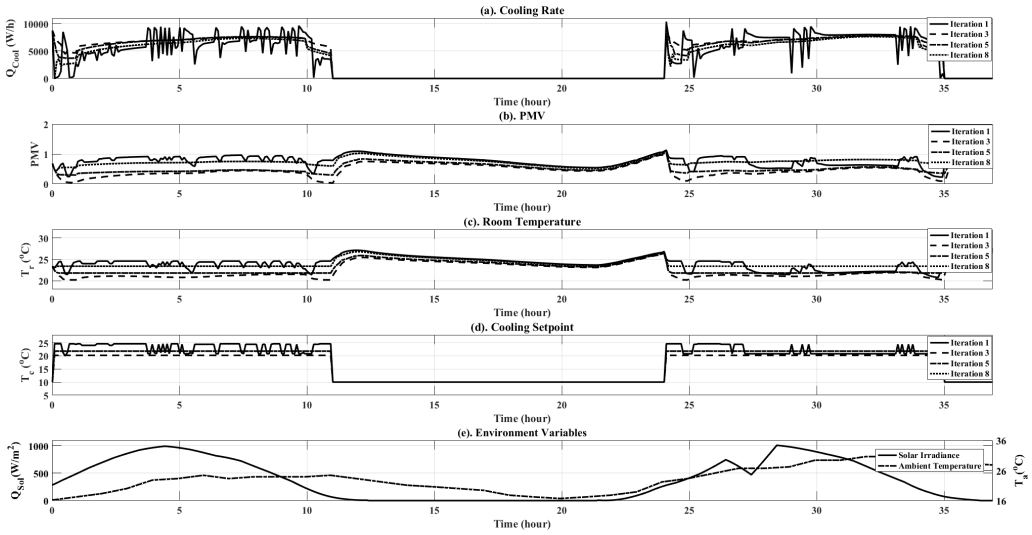

Figure 9. Simulation results plots for the RL controller at various iterations of training. Plots are shown for both objective variables (Q_{cool} and PMV), as well as additional simulation data, such as ambient temperature and solar irradiance.

Between the iterations, the aggression of the RL controller is tuned based on PMV and energy consumption metrics. This is evident from the steady decline in the standard deviation measure of cooling rate Q_{cool}, observed in Table 2. However, owing to the stochastic nature in which the neural networks are trained, there is a possible situation where the minimization goal is not met. In such a scenario, as observed in Table 3, Row 4, though an improved energy efficiency is evident, it greatly increases the PMV measure, indicating very poor thermal comfort. In such a scenario, the anomaly is discarded, and the next iteration is initialized with the previous valid iteration's outcome.

Table 3. Training comparison.

Control Type	PMV (Unitless)	Energy Total (W)
Ideal PMV	190.00	16,676
Variable Control	250.00	15,467
RL Control	211.3	16,298

A benchmarking of the performance of controllers under uniform environmental conditions is carried out. The experiments are performed using the B_{ideal} and $B_{variable}$ control structures.

B_{ideal} is a control that can offer an improved thermal comfort resulting from higher energy consumption. With the threshold provided in Equation (21), (τ = 0.75), the average value of the PMV, calculated in Equation (24), for the B_{ideal} case (0.66) can be relaxed further for improving energy efficiency without sacrificing the thermal comfort. A visualization is provided in Figure 10.

$$\tau = \frac{PMV_{Total}}{No.of Day \times No.of Samples/day} \tag{24}$$

This measure for the $B_{variable}$ case is 0.87. However, the RL controller achieves an average PMV of 0.73. This also improves the energy utilization by 2.27.

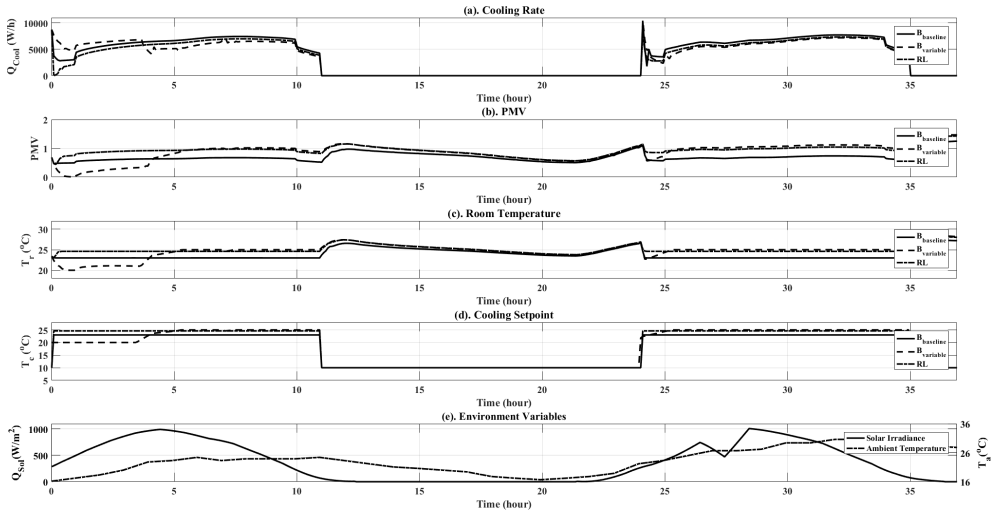

Figure 10. Simulation results plots for comparison between B_{ideal}, $B_{variable}$ and the RL controller over two days. Plots are shown for both objective variables (Q_{cool} and PMV), as well as additional simulation data, such as ambient temperature and solar irradiance.

The iterative tuning of the RL controller is performed on identical datasets. In order to evaluate the effectiveness of the RL controller, a five-day window, with weather data pertaining to five contiguous days, is utilized. These five days are not contiguous with respect to the training days; however, they are from the same season as the training data (summer), with slow-varying cloud cover and ambient temperature. The weather pattern provides cloud cover on Days 4 and 5, as observed in Figure 11 from the low solar irradiance and ambient temperature conditions. It is observed that the RL controller positions the cooling setpoint (T_c) slightly above the B_{ideal} controller. As observed from the cooling rate in Figure 11, it is evident that this enables a slightly lower cooling of the office thermal zone while leading to improved energy efficiency at the cost of acceptable thermal comfort loss. The average thermal comfort of the B_{ideal} is 0.708 and that of variable control is 0.917. The RL controller has an average thermal comfort of 0.775, improving the thermal comfort by 15.5% over the $B_{variable}$ and driving down energy by 5.03% as compared to the B_{ideal} controller.

Table 4 shows the numerical comparison between the RL controller and the two baselines. Due to the large variation in weather across the five-day validation phase, we can observe a more pronounced improvement from the RL controller. Compared to the ideal PMV baseline, the RL controller was able to achieve 2.12 kW of energy rate savings. Compared to the variable control baseline, the RL controller was able to maintain a 102 lower PMV total, which consumes only 1.44 kW more power. Similar to what is observed with the results from the training phase, the RL controller is able to sit at the sweet spot, where some slight thermal comfort is sacrificed for improvements in overall less HVAC power consumption.

Table 4. Validation comparison over 5 simulation days.

Control Type	PMV Total (Unitless)	Energy Total (W)
Ideal PMV	510	42,094
Variable Control	660	38,536
RL Control	558	39,978

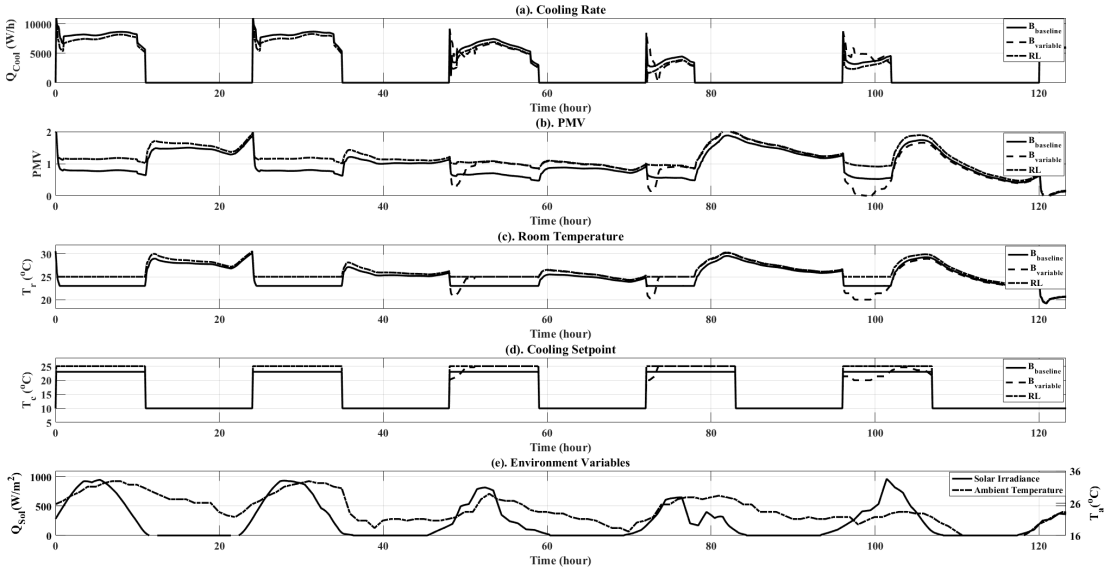

Figure 11. Simulation results plots for comparison between B_{ideal}, $B_{variable}$ and the RL controller over the five-day validation runs. Plots are shown for both objective variables (Q_{cool} and PMV), as well as additional simulation data, such as ambient temperature and solar irradiance.

7. Conclusions

A thermal zone of the office space has been designed. A platform for closed-loop simulation of HVAC systems has been created. Based on the influence of weather data on the building dynamics, uncertainties were introduced into the building model. A model-free reinforcement learning-based thermostat schedule controller has been developed using the novel long-short-term memory recurrent neural network, by closed-loop control of the HVAC system for two days. Using the simulation platform, we experiment on the B_{ideal} and $B_{variable}$ strategies. A five-day validation experiment is carried out with weather data from outside the training window. It is observed that during both training and validation, the RL controller was able to maintain the average PMV within an admissible range while saving energy compared to the B_{ideal} case. It provides the possibility to implement customized control for building HVAC control with minimal human intervention. Using the definition of the optimization cost, it will be possible to manipulate the goals of the RL controller towards improving the energy efficiency or thermal comfort.

To expand further upon the current simulation setup of a single zone, future research will be on developing RL thermostat schedule control across multiple non-ideal zones interacting with each other. In addition, integration with real building management systems (via Modbus or Lontalk) on the real-time building thermostat schedule will also be investigated.

Acknowledgments: This work is supported in part by the Vancouver International CleanTech Research Institute (VICTRI), Mitacs, Alberta Innovates and the Natural Sciences Engineering Research Council of Canada.

Author Contributions: Y.W. and K.V. conceived of and designed the experiments. Y.W. performed the experiments. Y.W., K.V. and B.H. analysed the data. Y.W. wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BCVTB	Building Controls Virtual Test Bed
BPTT	Back-Propagation Through Time
HVAC	Heating Ventilation and Air Conditioning
LSTM	Long-Short-Term Memory

PMV Predicted Mean Vote
RL Reinforcement Learning
RNN Recurrent Neural Network
TD Temporal Difference
VAV Variable Air Volume

References

1. International Energy Outlook 2016-Buildings Sector Energy Consumption—Energy Information Administration. Available online: https://www.eia.gov/outlooks/ieo/buildings.cfm (accessed on 25 April 2017).
2. Pérez-Lombard, L.; Ortiz, J.; Pout, C. A Review on Buildings Energy Consumption Information. *Energ. Build.* **2008**, *40*, 394–398.
3. EIA Issues AEO2012 Early Release—Today in Energy—U.S. Energy Information Administration (EIA). Available online: https://www.eia.gov/todayinenergy/detail.php?id=4671 (accessed on 25 April 2017).
4. Wallace, M.; McBride, R.; Aumi, S.; Mhaskar, P.; House, J.; Salsbury, T. Energy efficient model predictive building temperature control. *Chem. Eng. Sci.* **2012**, *69*, 45–58.
5. Široký, J.; Oldewurtel, F.; Cigler, J.; Prívara, S. Experimental analysis of model predictive control for an energy efficient building heating system. *Appl. Energ.* **2011**, *88*, 3079–3087.
6. Mirinejad, H.; Welch, K.C.; Spicer, L. A review of intelligent control techniques in HVAC systems. In Proceedings of the 2012 IEEE Energytech, Cleveland, OH, USA, 29–31 May 2012; pp. 1–5.
7. Jun, Z.; Kanyu, Z. A Particle Swarm Optimization Approach for Optimal Design of PID Controller for Temperature Control in HVAC. *IEEE* **2011**, *1*, 230–233.
8. Bai, J.; Wang, S.; Zhang, X. Development of an Adaptive Smith Predictor-Based Self-Tuning PI Controller for an HVAC System in a Test Room. *Energ. Build.* **2008**, *40*, 2244–2252.
9. Bai, J.; Zhang, X. A New Adaptive PI Controller and Its Application in HVAC Systems. *Energ. Convers. Manag.* **2007**, *48*, 1043–1054.
10. Lim, D.; Rasmussen, B.P.; Swaroop, D. Selecting PID Control Gains for Nonlinear HVAC & R Systems. *HVAC R Res.* **2009**, *15*, 991–1019.
11. Xu, M.; Li, S.; Cai, W. Practical Receding-Horizon Optimization Control of the Air Handling Unit in HVAC Systems. *Ind. Eng. Chem. Res.* **2005**, *44*, 2848–2855.
12. Zaheer-Uddin, M.; Tudoroiu, N. Neuro-PID tracking control of a discharge air temperature system. *Energ. Convers. Manag.* **2004**, *45*, 2405–2415.
13. Wang, Y.-G.; Shi, Z.-G.; Cai, W.-J. PID autotuner and its application in HVAC systems. In Proceedings of the 2001 American Control Conference (Cat. No.01CH37148), Arlington, VA, USA, 25–27 June 2001, pp. 2192–2196.
14. Pal, A.; Mudi, R. Self-tuning fuzzy PI controller and its application to HVAC systems. *Int. J. Comput. Cognit.* **2008**, *6*, 25–30.
15. Moradi, H.; Saffar-Avval, M.; Bakhtiari-Nejad, F. Nonlinear multivariable control and performance analysis of an air-handling unit. *Energ. Build.* **2011**, *43*, 805–813.
16. Anderson, M.; Buehner, M.; Young, P.; Hittle, D.; Anderson, C.; Tu, J.; Hodgson, D. MIMO robust control for HVAC systems. *IEEE Trans. Control Syst. Technol.* **2008**, *16*, 475–483.
17. Sahu, C.; Kirubakaran, V.; Radhakrishnan, T.; Sivakumaran, N. Explicit model predictive control of split-type air conditioning system. *Trans. Inst. Meas. Control* **2015**, doi:10.1177/0142331215619976.
18. Divyesh, V.; Sahu, C.; Kirubakaran, V.; Radhakrishnan, T.; Guruprasath, M. Energy optimization using metaheuristic bat algorithm assisted controller tuning for industrial and residential applications. *Trans. Inst. Meas. Control* **2017**, doi:10.1177/0142331217701538.
19. Dong, B. Non-linear optimal controller design for building HVAC systems. In Proceedings of the 2010 IEEE International Conference on Control Applications, Yokohama, Japan, 8–10 September 2010; pp. 210–215.
20. Greensfelder, E.M.; Henze, G.P.; Felsmann, C. An investigation of optimal control of passive building thermal storage with real time pricing. *J. Build. Perform. Simul.* **2011**, *4*, 91–104.
21. Kirubakaran, V.; Sahu, C.; Radhakrishnan, T.; Sivakumaran, N. Energy efficient model based algorithm for control of building HVAC systems. *Ecotoxicol. Environ. Saf.* **2015**, *121*, 236–243.

22. Bunin, G.A.; François, G.; Bonvin, D. A real-time optimization framework for the iterative controller tuning problem. *Processes* **2013**, *1*, 203–237.
23. Vaccari, M.; Pannocchia, G. A Modifier-Adaptation Strategy towards Offset-Free Economic MPC. *Processes* **2016**, *5*, 2.
24. Sutton, R.S.; Barto, A.G.; Williams, R.J. Reinforcement learning is direct adaptive optimal control. *IEEE Control Syst.* **1992**, *12*, 19–22.
25. Anderson, C.W.; Hittle, D.C.; Katz, A.D.; Kretchmar, R.M. Synthesis of reinforcement learning, neural networks and PI control applied to a simulated heating coil. *Artif. Intell. Eng.* **1997**, *11*, 421–429.
26. Barrett, E.; Linder, S. Autonomous HVAC Control, A Reinforcement Learning Approach. In *Machine Learning and Knowledge Discovery in Databases*; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2015; pp. 3–19.
27. Urieli, D.; Stone, P. A Learning Agent for Heat-pump Thermostat Control. In Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS '13), International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, USA, 6–10 May 2013; pp. 1093–1100.
28. Liu, S.; Henze, G.P. Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory: Part 2: Results and analysis. *Energ. Build.* **2006**, *38*, 148–161.
29. Liu, S.; Henze, G.P. Evaluation of reinforcement learning for optimal control of building active and passive thermal storage inventory. *J. Sol. Energ. Eng.* **2007**, *129*, 215–225.
30. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780.
31. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A search space odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, 1–11.
32. Fanger, P.O. Thermal comfort. Analysis and applications in environmental engineering. In *Thermal Comfort Analysis and Applications in Environmental Engineering*; Danish Technical Press: Copenhagen, Denmark, 1970.
33. Kober, J.; Bagnell, J.A.; Peters, J. Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274.
34. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533.
35. Giannoccaro, I.; Pontrandolfo, P. Inventory management in supply chains: A reinforcement learning approach. *Int. J. Prod. Econ.* **2002**, *78*, 153–161.
36. Ng, A.Y.; Jordan, M. PEGASUS: A policy search method for large MDPs and POMDPs. In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, Stanford, CA, USA, 30 June–3 July 2000; pp. 406–415.
37. Glynn, P.W. Likelihood ratio gradient estimation for stochastic systems. *Commun. ACM* **1990**, *33*, 75–84.
38. Sutton, R.S.; McAllester, D.A.; Singh, S.P.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *NIPS* **1999**, *99*, 1057–1063.
39. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv* **2015**, arxiv:1506.02438.
40. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NE, USA, 3–6 December 2012; pp. 1097–1105.
41. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
42. Chen, C.; Seff, A.; Kornhauser, A.; Xiao, J. Deepdriving: Learning affordance for direct perception in autonomous driving. In Proceedings of the IEEE International Conference on Computer Vision, Chile, 7–13 December 2015; pp. 2722–2730.