A Relearning Approach to Reinforcement Learning for control of Smart Buildings

Avisek Naug[1], Marcos Quiñones -Grueiro [2], and Gautam Biswas[3]

[1,2,3] *Vanderbilt University, Nashville, TN, USA*
avisek.naug@vanderbilt.edu
marcos.quinones@vanderbilt.edu
gautam.biswas@vanderbilt.edu

Abstract

This paper demonstrates that continual relearning of control policies using deep reinforcement learning (RL) can improve policy learning for non-stationary processes. We demonstrate this approach for a data-driven "smart building environment" that we use as a test-bed for developing HVAC controllers for reducing energy consumption of large buildings on our university campus. The non-stationarity in building operations and weather patterns makes it imperative to develop control strategies that are adaptive to changing conditions. On-policy RL algorithms, such as Proximal Policy Optimization (PPO) represent an approach for addressing this non-stationarity, but exploration on the actual system is not an option for safety-critical systems. As an alternative, we develop a RL technique that leverages a data-driven model of the system that is re-learnt periodically, and provides the basis for *policy relearning* by performing exploration on the re-learnt models to adapt to non-stationary behaviors. The relearning process is implemented in a way that avoids *catastrophic forgetting*. We compare the performance of our relearning RL controller to that of a static RL controller that does not implement the relearning function. The performance of the static controller diminishes significantly over time, but the relearning controller adjusts to changing conditions while ensuring comfort and optimal energy performance.

1. Introduction

Energy efficient control of Heating, Ventilation and Air Conditioning (HVAC) systems is an important aspect of building operations because they account for the major share of energy consumed by buildings. Most large office buildings,which are significant energy consumers, are structures with complex, internal energy flow dynamics and complex interactions with their environment. Therefore, building energy management is a difficult problem. Traditional building energy control systems are based on heuristic rules to control the parameters of the building's HVAC systems. However, analysis of historical data shows that such rule-based heuristic control is inefficient because the rules are based on simplified assumptions about weather and building operating conditions.

Recently, there has been a lot of research on *smart buildings* with smart controllers that sense the building state and environmental conditions to adjust the HVAC parameters to optimize building energy consumption (Shaikh et al., 2014). Model Predictive Control (MPC) methods have been successfully deployed for smart control (Maasoumy et al., 2014), but traditional MPC methods require accurate models to achieve good performance. Developing such models for large buildings may be an intractable problem (Smarra et al., 2018). Recently, Data-driven *MPC* based on random forest methods have been used to solve demand-response problems for moderate size buildings (Smarra et al., 2018), but is not clear how they may scale up for continuous control of large buildings.

Reinforcement Learning (RL) methods have recently gained traction for controlling energy consumption and comfort in smart buildings because they provide several advantages. Unlike MPC methods for robust receding horizon control, they have the ability to learn a locally optimal control policy without simulating the system dynamics over long time horizons. Therefore, while MPC methods require solving a non-linear optimization problem online, RL methods can directly compute the control action given the state space of the system because the control policy is learned offline (off policy algorithm) or learned online by compiling experiences over time (on policy algorithm) in a way that converges to optimal system behaviors. A number of reinforcement learning controllers for buildings have been proposed, where the building behavior under different environmental conditions are learnt from historical data (Mocanu et al., 2018; Naug et al., 2019). These approaches are classified as Deep Reinforcement Learning (DRL) approaches.

However, current data driven approaches for RL do not take into account the non-stationary behaviors of the building and its environment. Building operations and the environments in which they operate are continually changing, often in unpredictable ways. In such situations, the Deep RL controller performance degrades because the data that was used to train the controller becomes 'stale'. The solution to this problem is to detect changes in the building operations and its environment, and relearn the controller using data that is more relevant to the current situation. This paper proposes such an approach, where we relearn the controller at periodic intervals to maintain its relevance, and thus its performance.

The rest of the paper is organized as follows. Section 2 presents a brief review of some of the current approaches in model and data-driven reinforcement learning, and the concept of non-stationarity in MDPs. Section 3 introduce the theoretical problem of optimal control in a Non-Stationarity MDP. Section 4 formally introduces the relearning RL solution for non-stationary systems that we tackle in this paper and the associated theories behind the techniques used to solve the problem. Section 6 introduces the specific building for which we are trying to solve the problem and maps this building into the NS-MDP formulation proposed earlier. Section 7 then develops our data driven modeling as well as the incremental reinforcement learning schemes for 'optimal' building energy management for the specific building. Section 8 discusses our experimental results, and we finally present our conclusions and directions for future work.

2. LITERATURE REVIEW

Traditional methods for developing RL controllers have relied on accurate dynamic models of the system (model-based approaches) or data-driven approaches. We briefly review model-based and data-driven approaches for RL control, and then introduce the notion of non-stationary systems, and current RL work for non-stationary processes.

2.1. Reinforcement Learning with Model Based Simulators

Typical physics-based models of building energy consumption, use conservation of energy and mass to construct thermodynamic equations to describe system behavior. In (Wei et al., 2017), the authors applied Deep *Q-Learning* methods (Mnih et al., 2015) to optimize the energy consumption and ensure temperature comfort in a building simulated using EnergyPlus (Crawley et al., 2000), a whole building energy simulation program. In (Moriyama et al., 2018), the authors obtained cooling energy savings of 22% on an EnergyPlus simulated model of a data-center using a natural policy gradient based algorithm, TRPO (Schulman et al., 2015). Similarly, (Li et al., 2019) used an *off policy* algorithm called DDPG (Lillicrap et al., 2016) to obtain 11% cooling energy savings in an EnergyPlus simulation of a data-center. To deal with sample inefficiency in *on-policy learning*, (Hosseinloo et al., 2020) developed an event-triggered RL approach, where the control action changes when the system crosses a boundary function in the state space. They used a one-room EnergyPlus thermal to demonstrate their approach.

2.2. Reinforcement Learning with Data Driven Approaches

The examples above describe RL approaches applied to simple building architectures. As discussed, creating a model based simulator for large, complex buildings can be quite difficult (Park, 2013; Kim & Park, 2011). Alternatively, more realistic approaches for RL applied to large buildings rely on historical data from the building to learn data-driven models or directly use the data as experiences from which a policy is learnt. In (Nagy et al., 2018), the authors developed simulators from data-driven models and then used them for finite horizon control. Support Vector Regression was used in (Naug & Biswas, 2018) to develop a building energy consumption model, and then used stochastic gradient methods to optimize energy consumption. Authors in (Costanzo et al., 2016) used value-based neural networks to learn the thermodynamic model of a building. The energy models were then optimized using *Q-learning* (Sutton & Barto, 2018). Subsequently, (Naug et al., 2019) used a *DDPG* (Lillicrap et al., 2016) approach with a sampling buffer to develop a policy function that minimized energy consumption without sacrificing comfort. Another recent approach (Mocanu et al., 2018) has successfully applied deep RL to data-driven building energy optimization.

2.3. Non Stationary MDPs

The data-driven approaches presented in Section 2.2 do not address the non-stationarity of the large buildings. Non-stationary behaviors can be attributed to multiple sources. For example, weather patterns, though seasonal, can change abruptly and in unexpected ways. Similarly, conditions in a building can change quickly, e.g., when a large number of people enter the building for an event, or components of the HVAC system, degrade and fail, e.g, stuck valves, or failed pumps. When such situations occur, a RL controller, trained on the past experiences, cannot adapt to the unexpected changes in the system and environment, and, therefore, performs sub-optimally. Some methods (Mankowitz et al., 2018; Tamar et al., 2014; Shashua & Mannor, 2017) have been proposed to address non-stationarity in the environments by improving the *value function* under the worst case conditions (Iyengar, 2005) of the non-stationarity.

Other approaches try to minimize a *regret function* instead of finding the optimal policy for *non-stationary MDPs*. The regret function measures the sum of missed rewards when we compare the state value from a start state between current best policy and the target policy in hindsight *i.e.*, they tell us what actions would have been appropriate after the episode ends. This regret is then optimized to get better actions. For (Hallak et al., 2015), the authors applied this approach to context-driven MDPs (each context may represent a different non-stationary behavior) to find the piecewise stationary optimal policies for each context. They proposed a clustering algorithm to find a set of contexts. The papers (Jaksch et al., 2010; Gajane et al., 2019) also minimize the regret based on an average reward formulation instead of a state value function.In (Padakandla et al., 2019), the authors proposed a non-stationary MDP control method under a model-free setting by using a context detection method proposed in (Singh et al., 2019). These approaches assume knowledge of a known set of possible environment models beforehand, which may not be possible in real systems. Moreover, they are model-based, i.e., they assume the MDP models are available. Therefore, they cannot be applied in a model free setting.

To address non-stationarity issues in complex buildings we extend previous research in this domain to make the following contributions to data-driven modeling and RL based control of buildings:

- We retrain the dynamic behavior models of the building and its environment at regular intervals to ensure that the models respond to the distributional shifts in the system behavior, and, therefore, provide an accurate representation of the behavior.
- By not relearning the building and its environment model from scratch, we ensure the repeated training is not time consuming. This also has the benefit of the model not being susceptible to the catastrophic forgetting (Kirkpatrick et al., 2017) of the past behavior which is common in neural networks used for online training and relearning.
- We relearn the policy function; i.e., the HVAC controller every time the dynamic model of the system is relearned, so that it adapts to the current conditions in the building.

In the rest of this paper, we develop the relearning algorithms, and demonstrate the benefits of this incremental relearning approach on the controller efficiency.

3. OPTIMAL CONTROL WITH REINFORCEMENT LEARNING

Reinforcement learning (RL) represents a class of machine learning methods for solving optimal control problems, where an agent learns by continually interacting with an environment (Sutton & Barto, 2018). In brief, the agent observes the state of the environment, and based on this state/observation takes an action, and notes the reward it receives for the $(state, action)$ pair. The agent's ultimate goal is to compute a *policy*, i.e., a mapping from the environment states to the actions that maximizes the expected sum of reward. RL has been cast as a stochastic optimization method for solving Markov Decision Processes (MDPs), when the MDP is not known. We define RL problem more formally below.

Definition 3.1 (Markov Decision Process)**.** A Markov decision process is defined by a four tuple: $M = \{S, A, T, R\}$, where S represents the set of possible states in the environment. The transition function $T : S \times A \times S \rightarrow [0, 1]$ defines the probability of reaching state s' at $t + 1$ given that action $a \in A$ was chosen in state $s \in S$ at *decision epoch* t, $T = p(s'|s, a) = Pr\{s_{t+1} = s'|s_t = s, a_t = a\}$. The reward function $R : S \times A \rightarrow \Re$ estimates the immediate reward $R \sim r(s, a)$ obtained from choosing action a in state s.

The objective of the agent is to find an optimal policy π^* that maximizes the accumulated discounted rewards it receives over the future. The optimization criteria is the following:

$$V^{\pi^*}(s) = \max_{\pi \in \Pi} V^{\pi}(s) \ , \ \forall s \in S, \tag{1}$$

where $V^{\pi} : S \rightarrow R$ is called value function and it is defined as

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|s_0 = s\right] \ , \ \forall s \in S, \tag{2}$$

where $0 < \gamma \leq 1$ is called the discount factor, and it determines the weight assigned to future rewards. In other words, the weight associated with future rewards decays with time.

An optimal deterministic Markovian policy satisfying Equation 1 exists if the following conditions are satisfied

1. $|R \sim r(s, a)| \leq C < \infty, \forall a \in A, s \in S$
2. T and R do not change over time.

If a MDP satisfies the second condition, it is called a *stationary* MDP. However, most real world systems undergo changes that cause their dynamic model, represented by the transition function T, to change over time (Dulac-Arnold et al., 2019). In other words, these systems exhibit *non stationary behaviors*. Non stationary behaviors may happen because the components of a system degrade, and/or the environment in which a system operates changes, causing the models that govern the system behavior to change over time. In case of large buildings, the weather conditions can change abruptly, or changes in occupancy or faults in building components can cause unexpected and unanticipated changes in the system's behavior model. In other words, T is no longer invariant, but it may change over time. Therefore, a more realistic model of the interactions between an agent and its environment is defined by a non stationary MDP (NMDP) (Puterman, 2014).

Definition 3.2 (Non-Stationary Markov Decision Process)**.** A non-stationary Markov decision process is defined by a 5-tuple: $M = \{S, A, \mathcal{T}, (p_t)_{t \in \mathcal{T}}, (r_t)_{t \in \mathcal{T}}\}$. S represents the set of possible states that the environment can reach at decision epoch t. $\mathcal{T} = \{1, 2, ..., N\}$ is the set of decision epochs with $N \leq +\infty$. A is the action space. $p_t(s'|s, a)$ and $r_t(s, a)$ represent the transition function and the reward function at decision epoch t, respectively.

In the most general case, the optimal policy for a NMDP, π_t is also non stationary. The value of state s at decision epoch t within an infinite horizon NMDP is defined for a stochastic policy as follows:

$$V_t^{\pi}(s) = E\left[\sum_{i=t}^{\infty} \gamma^{i-t} R_i(s_i, a_i) | s_t = s,\ a_i \sim \pi_i,\ s_{i+1} \sim p_i\right] \tag{3}$$

Learning optimal policies from non-stationary MDPs is particularly difficult for non-episodic tasks when the agent is unable to explore the time axis at will. However, real systems do not change arbitrarily fast over time. Hence, we can assume that changes occur slowly over time. This assumption is know as the *regularity hypothesis* and it can be formalized by using the notion of Lipschitz Continuity (LC) applied to the transition and reward functions of a non-stationary MDP (Lecarpentier & Rachelson, 2019). This results in the definition of Lipschitz Continuous NMDP (LC-NMDP)

Definition 3.3 ((L_p, L_r) -LC-NMDP)**.** An (L_p, L_r) -LC-NMDP is a NMDP whose transition and reward functions are respectively L_p-LC and L_r-LC w.r.t. time,

$$W_1(p_t(.|s, a), p_{\hat{t}}(.|s, a)) \leq L_p|t - \hat{t}|,\ \forall\ (t, \hat{t}, s, s^{'}, a) \tag{4}$$

$$|r_t(s, a, s^{'}) - r_{\hat{t}}(s, a, s^{'})| \leq L_r|t - \hat{t}|,\ \forall\ (t, \hat{t}, s, s^{'}, a) \tag{5}$$

where W_1 represents the Wasserstein distance and it is used to quantify the difference between two distributions.

Although the agent does not have access to the true NMDP model, it is possible for the agent to learn a quasi-optimal policy by interacting with temporal slices of the NMDP assuming the LC-property. This means that the agent can learn using a stationary MDP of the environment at epoch t. Therefore, the trajectory generated by a LC-NMDP $\{s_0, r_0, ..., s_k\}$ is assumed to be generated by a sequence of stationary MDPs $\{MDP_{t_0}, ..., MDP_{t_0+k-1}\}$. In the next section, we present a continuous learning approach for optimal control of non stationary processes based on this idea.

4. CONTINUAL LEARNING APPROACH FOR OPTIMAL CONTROL OF NON-STATIONARY SYSTEMS

The proposed approach has two main steps: an initial offline learning process followed by continual learning process. Figure 1 presents the proposed approach organized in the following steps which are annotated as 1, 2 . . . in the figure:

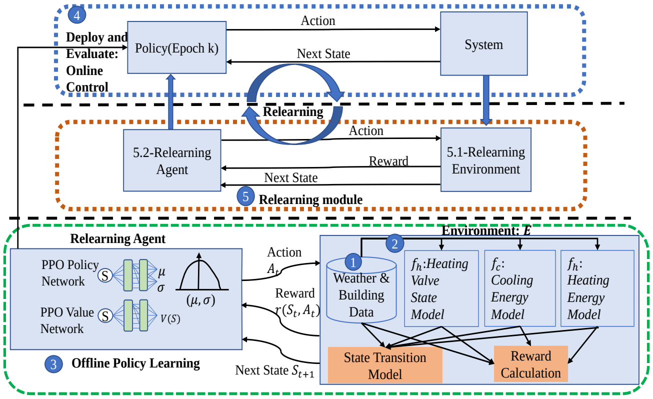

Figure 1. Schematic of our Proposed Approach

- Step 1. *Data collection*. Typically this represents historical data that may be available about system operations. In our work, we start with a data set containing information on past weather conditions and the building's energy-related variables. This data set may be representative of one or more operating conditions of the non stationary system, in our case, the building,
- Step 2. *Deriving a dynamic model of the environment*. In our case, this is the building energy consumption model, given relevant building and weather parameters.
  - A state transition model is defined in terms of state variables (inputs and outputs) and the dynamics of the system are learned from the data set.
  - The reward function used to train the agent is defined.
- Step 3. *Learning an initial policy*. A policy is learned offline by interacting with the environment model derived in the previous step.
- Step 4. *Deployment*. The policy learned is deployed online, i.e., in the real environment, and experiences from theses interaction are collected.
- Step 5. *Relearning*. In general, the relearning module would be invoked based on some predefined performance parameters, for example, when average accumulated reward value over small intervals of time is monotonically decreasing. When this happens:
  - the transition model of the environment is updated based on the recent experiences collected from the interaction with the up-to-date policy.
  - The current policy is re-trained offline, much like Step 3, by interacting with the environment now using the updated transition model of the system.

We will demonstrate that this method works if the regularity hypothesis is satisfied, i.e., the environment changes occur after sufficiently long intervals, to allow for the offline relearning step (Step 5) to be effectively applied. In this work, we also assume that the reward function, R, is stationary, and does not have to be re-derived (or re-learned) when episodic non stationary changes occur in the system.

Another point to note is that our algorithm uses a two-step *off line* process to learn a new policy: (1) learn the dynamic (transition) model of the system from recent experiences; and (2) relearn the policy function using the new transition model of the system. This approach addresses two important problems: (1) policy learning happens off line, therefore, additional safety check and verification methods can be applied to the learned policy before deployment $-$ this is an important consideration for *safety critical* systems; and (2) the relearning process can use an appropriate mix of past experiences and recent experiences to relearn the environment model and the corresponding policy. Thus, it addresses the *catastrophic forgetting* problem discussed earlier. This approach also provides a compromise between *off policy* and *on policy* learning in RL, by addressing to some extent the *sample inefficiency* problem.

We use Long Short-Term Memory (*LSTM*) Neural Network to model the dynamics of the system and the the Proximal Policy Optimization (*PPO*) algorithm to train the control policy. PPO is one of the best known reinforcement learning algorithm for learning optimal control law in short periods of time. Next, we describe our approach to modeling the dynamic environment using LSTMs, and the reinforcement learning algorithm for learning and relearning the building controllers (i.e., the policy functions).

5. LSTMs for Dynamic System Modeling and PPO for Policy Learning

5.1. Long Short-Term Memory Networks for Modeling Dynamic Systems

Despite their known success in machine learning tasks, such as image classification, deep learning approaches for energy consumption prediction have not been sufficiently explored (Amasyali & El-gohary, 2018). In recent work, Recurrent neural networks (RNN) have demonstrated their effectiveness for load forecasting when compared against standard Multi Layer Perceptron (MLP) architectures (Kong et al., 2019; Rahman et al., 2018). Among the variety of RNN architectures, Long-Short Term Memory (LSTM) networks have the flexibility for modeling complex dynamic relationships and the capability to overcome the so-called vanishing/exploding gradient problem associated with training the recurrent networks (Hochreiter & Schmidhuber, 1997). Moreover, LSTMs can capture arbitrary long-term dependencies, which are likely in the context of energy forecasting tasks for large, complex buildings.

The adaptive update of values in the input and forget gates provide LSTMs the ability to remember and forget patterns over time. The information accumulated in the memory cell is transferred to the hidden state scaled by the output gate. Therefore, training this network consists of learning the input-output relationships for energy forecasting by adjusting the weight matrices and bias vectors.

5.2. Proximal Policy Optimization

The Proximal Policy Optimization(*PPO*) algorithm (Schulman et al., 2017) has its roots in the Natural Policy Gradient method (S. M. Kakade, 2002), whose goal was to improve the common issues encountered in the application of policy gradients. Policy gradient methods(Sutton et al., 2000) represent better approaches to creating optimal policies, especially when compared to value-based reinforcement learning techniques. Value-based methods suffer from convergence issues when used with function approximators (Neural networks). Policy gradient methods also have issues with high variability, which have been addressed by Actor-Critic methods (Konda & Tsitsiklis, 2000). However, choosing the best step size for policy updates was the single biggest issue that was addressed in (Kakade & Langford, 2002). PPO replaces the log of action probability in the policy gradient equation

$$\mathrm{L}^{PG}(\theta) = \hat{\mathrm{E}}_t\Bigg[log\pi_\theta(a_t|s_t)\hat{A}_t\Bigg],$$

with the probability ratio $r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ inspired by (Kakade & Langford, 2002). Here, the current parameterized control policy is denoted by $\pi_\theta(a|s)$. $\hat{A}_t$ denotes the advantage of taking a particular action a compared to the average of all other actions in state s. According to the authors of PPO, this addresses the issue of the step size partially as they need to limit the values of this probability ratio. So they modify the objective function further to provide a *Clipped Surrogate Objective* function,

$$\mathrm{L}^{CLIP}(\theta) = \hat{\mathrm{E}}_t\Bigg[min(r(\theta)\hat{A}_t, clip(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\Bigg]. \tag{6}$$

The best policy is found by maximizing the above objective. The above objective has several interesting properties that makes PPO easily implementable and fast to reach convergence during each optimization step. The clipping ensures that the policy does not update too much in a given direction when the Advantages are positive. Also, when the Advantages are negative, the clipping makes sure that the probability of choosing those actions are not decreased too much. In other words, it strikes a balance between exploration and exploitation with monotonic policy improvement by using the probability ratio.

The PPO algorithm implements a parameterized policy $\pi_\theta(a|s)$ using a neural network whose input is the state vector S and the output is the mean μ and standard deviation σ of the best possible action in that state. The policy network is trained using the clipped objective function (see Equation 6) to obtain the best controller policy. A second neural network called the value network, $V(S)$, keeps track of the values associated with the states under this policy. This is subsequently used to estimate the advantage $\hat{A}_t$ of action A in state S. Its input is also S and its output is a scalar value indicating the average return from that state when policy $\pi_\theta(a|s)$ is followed. This network is trained using the Temporal Difference (*TD*) error (Sutton & Barto, 2018).

PPO has demonstrated to perform better than other gradient-based policy learning algorithms for a number of complex stochastic environment benchmarks, such as those provided in the *Mujoco* platform (Engstrom et al., 2019). One of our primary reasons for selecting the PPO algorithm is because it guarantees monotone improvement of the policy over multiple learning iterations. This property guarantees that the performance of the policy will not worsen during the relearning step described in the previous section.

6. PROBLEM FORMULATION FOR THE BUILDING ENVIRONMENT

We start with a description of our building environment and formulate the solution of the energy optimization problem by using our continuous RL approach. This section presents the dynamic data-driven model of building energy consumption and the reward function we employ to derive our control policy.

6.1. System Description

The system under consideration is a large three-storeyed building on our university campus. It has a collection of individual office spaces, classrooms, halls, a gymnasium, a student lounge, and a small cafeteria. The building climate is controlled by a combination of Air Handling Units(*AHU*) and Variable Refrigerant Flow (*VRF*) systems (Naug et al., 2019). The configuration of the HVAC system is shown in Figure 2.

The AHU brings in fresh air from the outside and adjusts the air's temperature and humidity before releasing it into the building. Typically, the desired humidity level in the building is set to 50%, and the desired temperature values are set by the occupants. Typically, the air is released into the building at a neutral temperature (usually 65^oF or 72^oF). The VRF units in the different zones further heat or cool the air according to the respective temperature set-point (defined by the occupants' preferences). The AHU has two operating modes depending on the outside wet bulb temperature. When the wet bulb temperature is above 52^oF, only the cooling and the reheat coils operate. The AHU dehumidifies the air using the cooling coil to reduce the air temperature to 52^oF, thus causing a condensation of the excess moisture, and then heats it back up to a specific value that was originally determined by a rule-based controller (either 65^oF or 72^oF). When the wet bulb temperature is below 52^oF (implying the humidity of the outside air is below 50%), only the preheat coil operates to heat the incoming cold air to a predefined set-point. The discharge temperature (reheating and preheating set-point depending on the operating mode).

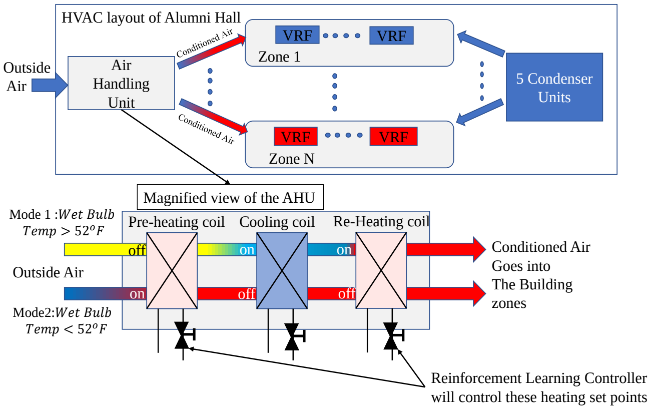
HVAC layout of Alumni Hall
Outside Air
Air Handling Unit
Conditioned Air
Conditioned Air
VRF
VRF
Zone 1
VRF
VRF
Zone N
5 Condenser Units
Magnified view of the AHU
Mode 1 :Wet Bulb Temp > 52°F
Outside Air
Mode2:Wet Bulb Temp < 52°F
Pre-heating coil
Cooling coil
Re-Heating coil
off
on
on
on
off
off
Conditioned Air Goes into The Building zones
Reinforcement Learning Controller will control these heating set points

Figure 2. Simplified schematic of the HVAC system under Study

6.2. Problem Formulation

The goals of our RL controller is to determine the discharge air temperature set-point of the AHU to minimize the total heating and cooling energy consumed by the building without sacrificing comfort. We will formulate the RL problem by specifying the state-space, the action-space, the reward function, and the transition function for the our building environment.

6.2.1. State Space

The overall energy consumption of our building depends on how the AHU operates but also on exogenous factors such as the weather variability and the building occupancy. The evolution of the weather does not depend on the state of the building. Following Diettrich et al (2015), we formulate our control problem as a non-stationary exogenous state MDP. The latter can be formalized as follows:

Definition 6.1 (Exogenous State Markov Decision Process)**.** An Exogenous State Markov decision process is defined by a Markov Decision Process which transition function satisfies the following property

$p(e', x'|e, x, a) = Pr(x_{t+1} = x'|x_t = x)Pr(e_{t+1} = e'|e_t = e, x_t = x, a_t = a)$,

where the state space of the MDP is divided into two sub-spaces $S = \mathbf{X} \times \mathbf{E}$ such that $x \in \mathbf{X}$ and $e \in \mathbf{E}$. $\mathbf{X}$ is the set of exogenous state variables whose evolution do not depend on the MDP Actions. $\mathbf{E}$ is the set of endogenous state variables whose next state depends on the action a_t.

The above definition combined with non-stationarity creates time dependency, where the transition function can change from epoch to epoch as the system behavior evolves. For our building, the subset of exogenous variables of the subspace $\mathbf{X}$ are: (1) Outside Air Temperature (*oat*), (2) Outside Air Relative Humidity (*orh*), (3) Wet Bulb Temperature (*wbt*), (4) Solar irradiance (*sol*), (5) Average Building Temperature Preference Set Point (*avg-stpt*). The remaining variables corresponding to the subspace $\mathbf{E}$ are (6) AHU Supply Air Temperature (*sat*), (7) Heating energy for the Entire Building(f_h) and (8) Cooling energy for the Entire Building (f_c). Since building occupancy is not measured at this moment, we cannot incorporate that variable to our state space.

6.2.2. Action Space

The action space A_t of the MDP in each epoch $\mathcal{T}$ is the change in the neutral AHU Supply Air Temperature Set-Point(*sat-stpt*). As discussed before, the wet bulb temperature determines the AHU operating mode. The valves and actuators that operate the HVAC system have a certain latency in their operation. This means that our controller must not arbitrarily change the Supply Air Temperature Set-Point directly. We, therefore, adopt a safe approach where the action space A is defined as a continuous variable $\in [-2^oF, +2^oF]$ that represents the range in which the Supply Air Temperature Set Point can be changed from one time step to the next. In this work, we set Δ_t, the time step to be 30 minutes, given the slow dynamics of the building.

Based on the above action, the AHU Supply Air Temperature Set Point(*sat-stpt*) for next time step(t+1) is updated from its previous value at t as

$$\textit{sat-stpt}_{t+1} = \textit{sat-stpt}_t + A_t \tag{7}$$

Note that this Supply Air Temperature Set Point(*sat-stpt*) will determine the actual AHU Supply Air Temperature(*sat*) as discussed in the next section.

6.3. Transition Model

Taking into consideration that the state and action space of the building are continuous, the transition function will comprise three components.

First, the transition function of the exogenous state variables $Pr(x'|x)$ is not explicitly modeled (*oat*, *orh*, *wbt*, *sol*, and *avg-stpt*). Their next state (x_{t+1}) at time $t+1$ is provided by a weather database. These variables are available at 5 minute intervals through a Metasys portal for most buildings on our campus; solar irradiance, *sol*, is acquired from external data sources.

The AHU Supply Air Temperature(*sat*), the heating and cooling energies are the endogenous state variables since their behavior at $t+1$ depends on the supply air temperature set point at t. We assume that the supply air temperature at the next time step (sat_{t+1}) will change according to the supply air temperature set point ($\textit{sat-stpt}_t$) as follows

$$sat_{t+1} = \textit{sat-stpt}_t + \eta \tag{8}$$

where η represents the error of the Proportional-Integral controller that governs the supply air temperature control loop. We characterized that error from historical data to be $\eta \sim \mathcal{N}(0, 0.002)$.

Lastly, the heating and cooling energy variables(f_h and f_c) are determined by the transition functions

$$f_{h,t+1} = F(\mathbf{X}_t, f_{h,t}, sat_t), \tag{9}$$

$$f_{c,t+1} = F(\mathbf{X}_t, f_{c,t}, sat_t), \tag{10}$$

where $\mathbf{X}_t = \left[oat_t, orh_t, wb_t, sol_t, \textit{avg-stpt}_t\right]$. $f_{h,t}$ and $f_{c,t}$ are the corresponding heating and cooling energy values at the current time step. As discussed in the last section, we train stacked LSTMs to derive nonlinear approximators of these functions.

6.4. Reward Function

The reward function includes two components: (1) the total energy savings for the building expressed as heating and cooling energy savings, and (2) the comfort level achieved. The reward signal at time instant t is given by

$$r_{t+1}(S_t, A_t) = \vartheta * Reward_{energy} + (1-\vartheta) * Reward_{comfort} \tag{11}$$

where $\vartheta \in [0, 1]$ defines the importance we give to each term. We considered $\vartheta = 0.5$ in this work.

$Reward_{energy}$ is defined in terms of the energy savings achieved with respect to the previous rule-based controller for the building, *i.e.* we reward the RL controller when its actions result in energy savings calculated as the difference between the total heating and cooling energy under the RBC controller actions and the RL controller actions. $Reward_{energy}$ is defined as follows

$$\begin{aligned} Reward_{energy} = {} & RBC_{valve,t} * RBC_{heating,t} \\ & - RL_{valve,t} * RL_{heating,t} + RBC_{cooling,t} \\ & - RL_{cooling,t} \end{aligned} \tag{12}$$

where the components of this equation are

- $RL_{heating,t}$: The total energy used to heat the air at the heating or preheating coil as well as the VRF system over the last time interval(Δ_t) ending at t based on the heating set point at the AHU assigned by the RL controller.
- $RBC_{heating,t}$: The total energy used to heat the air at the heating or preheating coil as well as the VRF system over the last time interval(Δ_t) ending at t based on the

heating set point at the AHU assigned by the Rule Based Controller(*RBC*).

- $RL_{valve,t}$: The on-off state of the heating valve at time-instant t based on the heating set point at the AHU assigned by the RL controller.
- $RBC_{valve,t}$: The on-off state of the heating valve at time-instant t based on the heating set point at the AHU assigned by the Rule Based Controller(*RBC*).
- $RL_{cooling,t}$: The total energy used to cool the air at the cooling coil as well as the VRF system over the last time interval(Δ_t) ending at time-instant t based on the set point at the AHU assigned by the RL controller.
- $RBC_{cooling,t}$: The total energy used to cool the air at the cooling coil as well as the VRF system over the last time interval(Δ_t) ending at time-instant t based on the set point at the AHU assigned by the Rule Based Controller(*RBC*).

Here by Rule Based Controller set-point, we refer to the historical set point data that is obtained from the past data that we run our comparisons against.

The heating and the cooling energy are calculated as a function of the exogenous state variables $\mathbf{E}_{t+1}$ and A_t, as discussed in the previous sub-section. Additionally, we model the behavior of the valve that manipulates the steam flow in the coil of the heating system, This valve shuts off under certain conditions causing the heating energy consumption to drop to 0. This hybrid on-off behavior is not easily modeled by a LSTM thus we need to model the valve behavior independently as an on-off switch to decide when to consider the predictions made by the LSTM (only during on). Note that both $RBC_{valve,t}$ and $RL_{valve,t}$ are predicted by using a binary classifier.

For this paper, the reward associated with comfort is measured at a gross level, i.e., by how close the AHU Supply Air Temperature(*sat*) is to the Average Building Temperature Preference set-point(*avg-stpt*) at any time instant. Let $\delta_t = abs(\textit{avg-stpt} - sat)$

$$Reward_{comfort} \begin{cases} \frac{1}{\delta_t+1}, & if \;\; \delta_t \leq 10^oF \\ -\delta_t, & if \;\; \delta_t > 10^oF \end{cases} \tag{13}$$

The comfort term allows the RL controller to explore in the vicinity of the average building temperature preference to optimize energy. The 1 added to the denominator in case 1 makes the reward bounded.

The overall reward space is non-sparse so the RL agent would have sufficient heuristic information for moving towards an optimal policy as it explores the environment.

7. IMPLEMENTATION

In this section, we describe the implementation of the proposed approach for the optimal control of the system described in the previous section.

7.1. Data Collection and Processing

For Step 1 (see Figure 1) the energy data from the building was collected over a period of 20 months(July 2018 to Feb 2020) using the *BACNET* system. These include the weather variables, the building temperature set points, and building energy consumption available at 5 minute intervals. First, we cleaned the data by removing the statistical outliers(two standard deviations approach). Next, we aggregated the variables into half-an-hour intervals using an aggregation and averaging processThen we scaled the data using a Min-Max approach to the $[0, 1]$ interval so that we can learn the different data-driven models and the controller policy. In order to perform the off-line learning as well as the subsequent relearning, we sampled this above data in windows of 3 months (for training) and 1 week (for evaluating).

7.2. Deriving the Energy and Valve Models

In this section we derive the two building energy consumption models, and the valve model that governs the heating energy consumption by the building. This is step 2 in Figure 1.

The general neural network architecture for deriving the data driven energy models included: (1) Feed Forward Neural Networks(Mao & Jain, 1995) to model the non-linear relationships between the inputs variables and generate a rich set of features; and (2) input the derived features to a LSTM model and derive the energy consumption dynamics of the building(Mohajerin & Waslander, 2019; Zheng et al., 2017). The number of layers and units for each type of network were determined by performing *hyper-parameter optimization* over a wide range of possible values for these parameters.

The strategy to set the hyper-parameters of data-driven methods is important since the overall performance might depend on them. Manually trying to determine the hyper-parameters is intractable since there are possible combinations of the variables is infinitely large. Therefore, we use Bayesian optimization to adjust the hyper-parameters of the data-driven models (Snoek, Larochelle, & Adams, 2012). Bayesian optimization is a sequential approach for global optimization that does not require calculation of the derivatives of the function to be optimized.

7.2.1. Heating Energy model

The heating energy model model derives the heating energy consumption at $t + 1$ as a function of the heating energy consumed and the controller action taken at time t. The model for Heating energy f_h is learned from a sequence of variables comprising the states S_{t+1} over the last 3 hours *i.e.* 6 samples considering data samples at 30 minute intervals. The output for the heating energy model is the total historical heating energy consumed over next 30 minute interval.

The heating coils for the building operate in a hybrid mode where the heating valve is turns on and off to (a) supply heat for the pre-heating mode, and (b) supply heat to the VRF systems to heat the building if that cannot be achieved by the cold water loop. When the valve is turned off, we assume that the heating energy consumed is 0. This abrupt change cannot be modeled by a smooth LSTM model. Therefore, we train our model on contiguous sections when the heating valve is on.

The model for f_h is constructed by stacking six Fully Feed Forward Neural (*FFN*) Network Layers of sixteen units each followed by two layers of LSTM with four units each. The activation for each layer is *Relu*. The FFN layers generate the rich feature set from the input data and the LSTM layers learn the nonlinear function that approximates the dynamics of the energy consumption of heating energy system.. The learning rate is initially set to 0.001, and is changed according to a linear schedule to ensure faster changes at the beginning followed by gradual changes near the optimum to avoid overshooting and oscillations. A mean squared error calculation on validation data is used to determine the terminating condition. The best model parameters and network architectures were found by hyper-parameter tuning via Bayesian Optimization on a Ray-Tune (Liaw et al., 2018) cluster.

7.2.2. Valve State model

The valve model f_v models the on-off for the heating energy, i.e., when the heating energy consumption is non zero and zero. The input variables to this model are the same as the Heating Energy model and the output is the valve (heating coil) on-off state.

The model for f_v is constructed by stacking four Fully Feed Forward Layers of sixteen units each followed by two layers of LSTM with eight units each. The activation for each layer is *Relu*. The learning rate, validation data, and the model hyper-parameters are similarly chosen as before. The loss used in this case is the binary cross-entropy loss since it is a two-class prediction problem.

7.2.3. Cooling Energy model

The cooling energy model is used to calculate the cooling energy consumed in state S_{t+1} when the action A_t is taken in state S_t. The input to this model is the same as the Heating Energy model. The output of the model is the cooling energy consumed over the next 30 minute interval.

The model for f_c is constructed by stacking six Fully Feed Forward Layers of sixteen units each followed by two layers of LSTM with eight units each. The activation for each layer is *Relu*. The learning rate, validation data and the model hyper-parameters are chosen in a way similar to the Heating Energy Model.

Once the two energy models and the valve model have been learned, we construct the data-driven simulated environment E. It receives the control action A_t from the PPO controller and generates the next state S' from the current state, S. To calculate S', the weather values for the next state are obtained by simple time-based lookup from the "Weather Data" database. The supply air temperature for the next state is obtained from the "State Transition Model" using Equation 8. The reward $r_{t+1}(S, A_t)$ is calculated using Equation 11. Every time the Environment is called with an action, it will perform this entire process and return the next state S', the reward $r_{t+1}(S_t, A_t)$ back to the RL controller with some additional information on the current episode.

7.3. PPO Controller

As discussed previously in section 5.2, the controller will learn two neural networks using the feedback it receives from the environment E in response to its action A_t. This action is generated by sampling from a normal distribution with mean μ and standard deviation σ which are the outputs of the PPO policy network as shown in figure 1. After sampling responses from the environment for a number of times, the collected experiences under the current controller parameters, are used to update the controller network by optimizing $\mathrm{L}^{CLIP}(\theta)$ in Equation 6 and the value networks by TD Learning. We repeat this training process until the optimization has converged to a local optima.

The Policy Network model architecture consists of two layers of Fully Feed Forward Layers with 64 units each. The Value Network network structure is identical to the Policy network. The networks are trained on-policy with a learning rate of 0.0025. Each time the networks were trained over 10^6 steps through the environment. For the environment E this corresponded to approximately 3000 episodes where each episode was a week long.

7.4. Evaluating the energy models, valve models, and the PPO controller

This corresponds to Step 4 in Figure 1. Once the energy model, the valve state models, and the controller training have converged we evaluate them on a held out test data for 1 week. The Energy models are evaluated using the Coefficient of variation Root Mean Square Error (*CVRMSE*)

$$CVRMSE = \frac{\sqrt{\sum_i (y_{true} - y_{pred})^2}}{\bar{y_{true}}} \tag{14}$$

where y_{true} and y_{pred} represent the true and the predicted value of the energy, respectively.

The valve model is evaluated based on its ROC-AUC as the on-off dataset was found to be imbalanced. The controller policy is evaluated by comparing the energy savings for the cooling energy and the heating energy as well as how close the controller set-point for the AHU Supply Air Temperature(*sat*) is to the building average set-point(*avg-stpt*).

7.5. Relearning Schedule

Steps 4 and 5 in Figure 1 are repeated by moving the data collection window forward by 1 week. We observed that having a large overlap over training data between successive iterations helps the model retain previous information (*i.e.*, avoid catastrophic forgetting), while gradually adapt to the changing data.

From the second iteration onward we do not train the data driven LSTM models (*i.e.* f_h, f_v, f_c) from scratch. Instead, we use the pre-trained models from the previous iteration to start learning on the new data. For the energy models and valve models we no longer train the FFN layers and only retrain the head layers comprising the LSTMs. The FFN layers are used to learn the representation from the input data and this learning is likely to stay identical(we also observed this empirically from our experiments) for different data. The LSTM layers, on the other hand, model the trend in the data which must be relearnt due to the distributional shift. Our results show that this training approach saves time with virtually no loss in model performance. We also adapt(retrain) the pre-trained PPO controller policy network according to the changes in the system. This continual learning approach saves us time during repeated retraining and allows the data-driven models and the controller to adapt to the non-stationarity of the environment.

8. RESULTS

This section illustrates the performance of our energy models, valve model, and the RL controller over multiple weeks.

8.1. Relearning Results for Heating Energy Model

Figure 3 shows the heating energy prediction on a subset of the data from October 7th to 23rd. We selected this time period because the effects of the non-stationarity in the data were quite apparent. We compare the prediction of a fixed model, which was not updated after October 7th, with a model which was retrained by including the new week's data from 7th to the 13th. The figure demonstrates the necessity of relearning the heating energy model at regular intervals. After the October 12th, the AHU switches from using the reheating to the preheating coil due to colder weather as indicated by the wet bulb temperature. This causes the heating energy consumption to change abruptly. The model which was not updated after October 7th cannot learn this behavior and its predictions based on the initial model, show increasing deviations with time. On the other hand, the weekly relearning model behavior started degrading but once it was retrained using the data from Oct 7th to the 13th, it captured the changing behavior quickly using a small section of similar data in its training set. The overall CVRMSE for the relearning energy model is shown in Figure 4. For majority of the weeks, the CVRMSE is below 30% which is accepted according to ASHRAE guidelines for energy prediction at half hour intervals

8.2. Relearning Results for Cooling Energy Model

Figure 5 shows the plots for predicting the Cooling energy Energy over a span of two weeks. We also include the the energy prediction from a fixed model. Starting from 25th April, both the Fixed and Relearning model for Cooling Energy predictions start degrading as they start following an increasing trend while the actual trend is downward and this behavior is expected while learning on non-stationary data. But the Relearning Cooling Energy model is retrained using the data from April 19th to April 26th at the end of the week corresponding to 26th April. Thus its predictions tend to be better than a fixed model for the next week whose predictions degrade as the week progresses.The overall CVRMSE from week to week for the relearning energy model is shown in Figure 6. For all the weeks, the CVRMSE is below 30% which is an acceptable error rate according to ASHRAE guidelines for energy prediction at half hour intervals

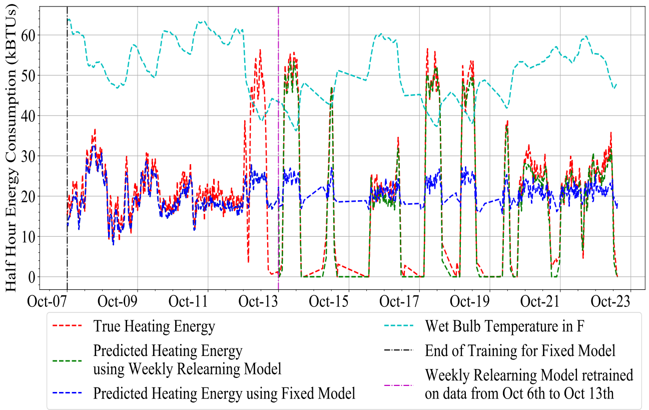

Figure 3. Comparison of true versus predicted Heating Energy for a weekly relearning model and a static/non-relearning model

8.3. Prediction of the Heating Valve status

Figure 8 shows the Area Under the Receiver Operating Characteristics (ROC AUC) for the model predicting the valve status(on/off). We also show the actual and predicted valve state for around one month in Figure 7. Overall, the relearning

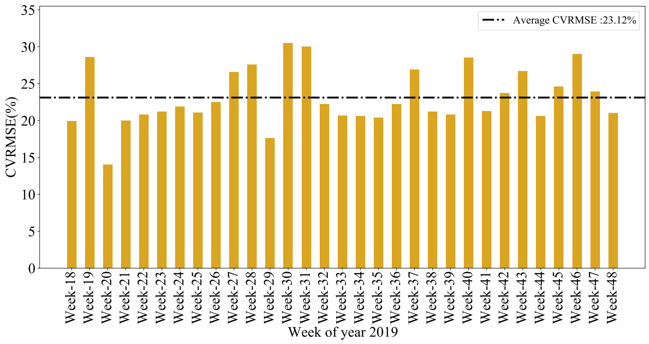

Figure 4. The weekly CVRMSE of the Hot Water Energy Relearning Model for predicting Hot Water Energy consumption at half hour intervals

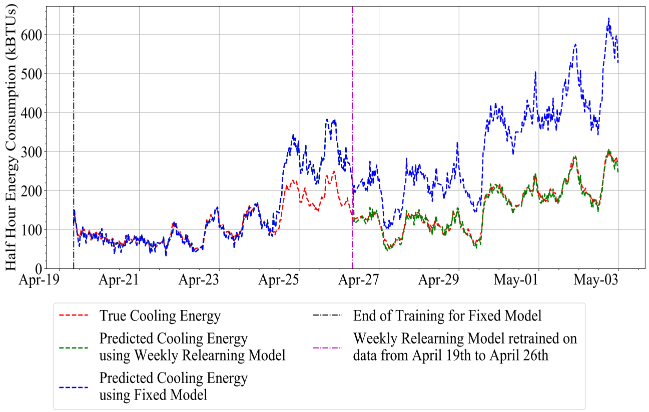

Figure 5. Comparison of true versus predicted Cooling Energy for a weekly relearning model and a static/non-relearning model

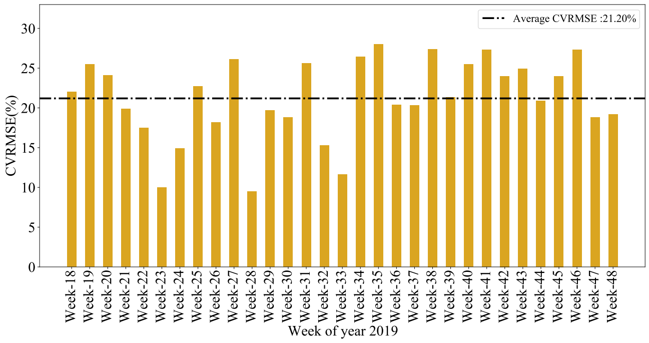

Figure 6. The weekly CVRMSE of the Cooling Energy Relearning Model for predicting Cooling Energy consumption at half hour intervals

valve model is able to accurately predict the valve behavior with an average week by week accuracy of 88.62%.

8.4. Training Episode Reward

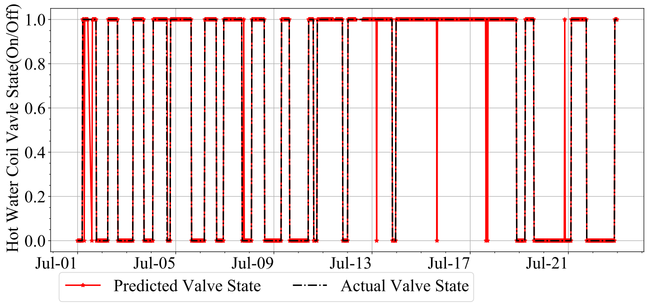

Figure 7. Comparing True versus Predicted Hot Water Valve State behavior

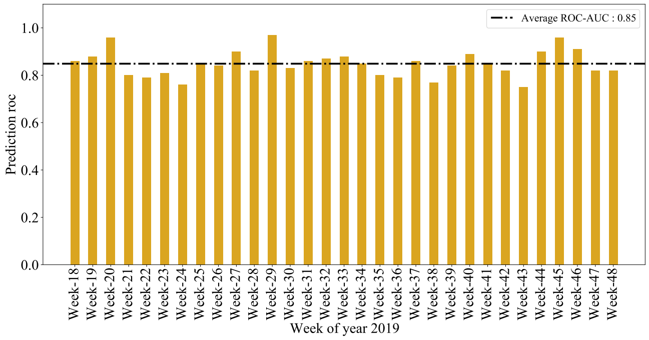

Figure 8. Hot Water Valve State Prediction model ROC AUC evaluated over multiple weeks

We trained the PPO controller on the environment E every week to adjust to the non stationarity of our system. The episode-wise cumulative reward metric from Equation 11 is used to asses the improvement in controller performance over a number of weeks. We observed that even though the controller was able to achieve good results after training over a couple of weeks of data, it keept improving as weeks progressed. The cumulative reward metric is plotted in Figure 9. The occasional drops in the average reward were due to changing environment conditions as the training progressed.

8.5. Cooling Energy Performance

We compared the cooling energy performance of both the adaptive reinforcement learning controller and a static reinforcement learning controller against a rule based controller.

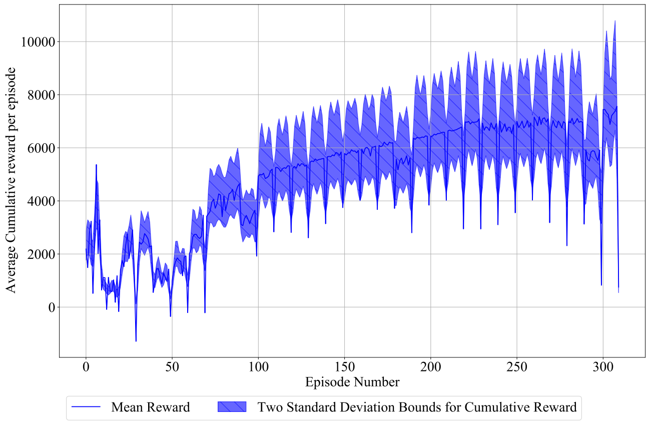

Figure 9. Average Cumulative Reward Obtained across each episode trained across 10 environments in parallel

A plot comparing the cooling energy consumed over a segment of the evaluation period is shown in Figure 10. The significance of the relearning is apparent in this segment of the data. When we calculate the energy savings for each RL controller, the static RL controller had slightly higher cooling energy savings because the last version of it was trained during warmer weather and it tends to keep the building cooler. But when the outside temperature drops, the static controller action does not heat the system too much resulting in the VRF systems starting to heat the building which consume higher energy. The cooling energy savings over the period shown in figure 10 was 9.3% for the adaptive controller and 11.2% for the static controller. The average weekly cooling energy savings over the entire evaluation period of 31 weeks was $12.61\%(5.73\%)$ or $188.53(18.153)$ kBTUs for the adaptive controller versus $12.81\%(8.22\%)$ or $191.21(23.009)$ kBTUs for the non-adaptive/static controller.

8.6. Heating Energy Performance

Similarly, we compared the heating energy performance of our relearning versus the static controller over the same timeline as shown in Figure 11. This plot shows the severe issue of over-cooling that can occur in the building when the controller is not updated regularly. Due to lower Supply Air Temperature Set Point of the static controller, the total heating energy consumption for the building went up over the entire period of cool weather. The heating energy savings over the period shown in Figure 11 was 6.4% for the adaptive controller while the static controller increased the energy consumption by 65%. The average weekly heating energy savings over the entire evaluation period of 31 weeks was $7.19\%(2.188\%)$ or $112.19(13.91)$ kBTUs for the adaptive controller whereas the con-adaptive/static controller increased the energy consumption by $54.88\%(32.66\%)$ or $161.08(18.211)$ kBTUs.

The sum total of the heating and cooling energy consumption under the historical rule based controller, the adaptive controller and the non-adaptive/static controller is shown in Figure 12. The adaptive controller consistently saved more energy than the non-adaptive controller. Overall the adaptive controller was able to save 300.72 kBTUs each week on average whereas the static controller was able to save only 30.03 kBTUs.

8.7. Control Actions

Here we show why the overall energy consumption of the building went up when we use a static reinforcement learning controller. We plot the Discharge/Supply Air Temperature set-point resulting from the actions of both the adaptive and static controller along with outside air temperature and relative humidity in Figure 13. On October 12th, the outside temperature went down and both the adaptive and static controller failed to improve building comfort condition. After October 13th , the adaptive controller was re-trained by considering the last week's data where it encountered environment states with lower outside air temperatures and it adapted to those conditions. For the remaining time period analyzed, the adaptive controller kept the Discharge/Supply Air Temperature set-point closer to the comfort conditions required by the occupants.

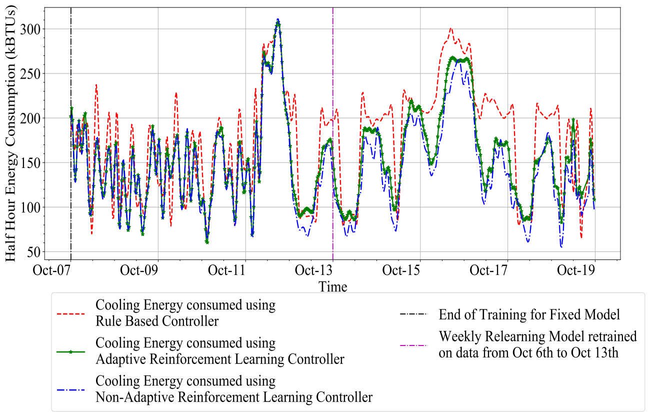

Figure 10. Plot of Cooling Energy Consumed for actions based on RBC, Adaptive RL controller and Static RL Controller

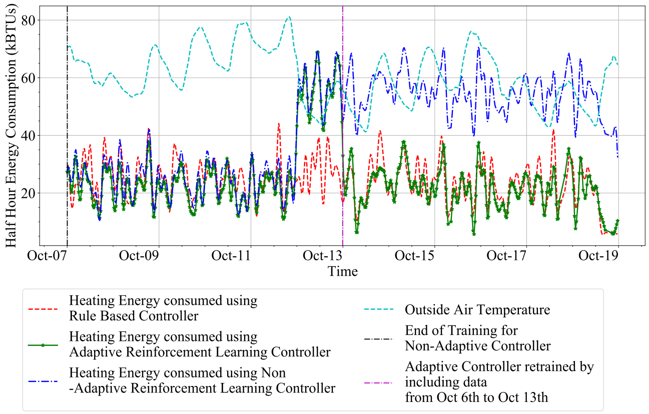

Figure 11. Plot of Heating Energy Consumed for actions based on RBC, Adaptive RL controller and Static RL Controller

How can we ensure that the increasing performance of the controller is due to the relearning approach and not just because an increasing number of training samples are being used? We observe that the data from the past that was used to train the controller in previous episodes, is no longer relevant when the new control policy is computed. The addition of the recent experiences during the relearning phase helps us adapt to the new behavior of the system.

CONCLUSIONS

In this paper, we have presented the design and implementation of a Relearning controller, and demonstrated through

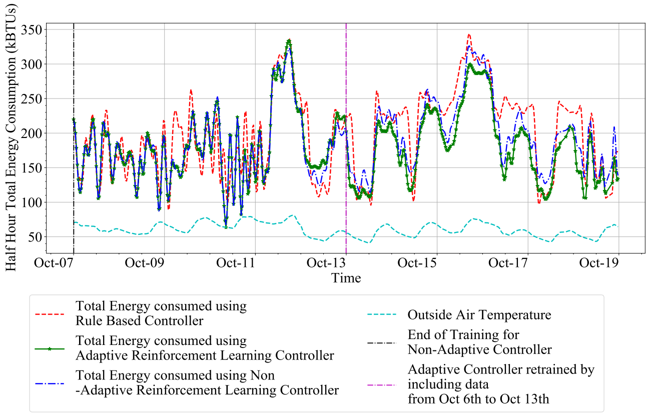

Figure 12. Plot of Total Energy Consumed for actions based on RBC, Adaptive RL controller and Static RL Controller

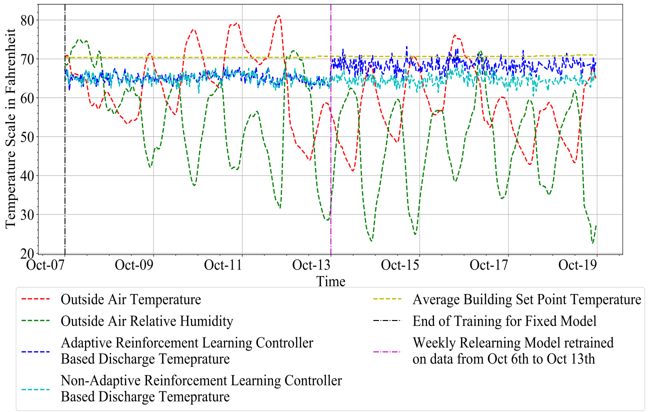

Figure 13. Plot of Supply Air setpoint(sat) based on actions chosen by the Adaptive RL controller versus Static RL Controller

a case study, how this controller results in improved energy savings over a static RL controller as well as the baseline, which is the original rule-based controller for the building. For safety and practical reasons, we capped the discharge temperature set point changes to $\pm 2^o C$. Whereas this may have led to somewhat reduced energy savings, it ensured that our controller was stable, and it would not generate spurious settings that significantly affects occupant comfort.

In future work, we plan to develop more intelligent relearning methods, for example, where we relearn the model and controller only when the controller performance starts degrading. Switching from fixed interval to on-demand epochs, will result in more computationally efficient implementations for online smart building control.

REFERENCES

Amasyali, K., & El-gohary, N. M. (2018). A review of data-driven building energy consumption prediction studies. *Renewable and Sustainable Energy Reviews*, *81*, 1192–1205.

Costanzo, G. T., Iacovella, S., Ruelens, F., Leurs, T., & Claessens, B. J. (2016, jun). Experimental analysis of data-driven control for a building heating system. *Sustainable Energy, Grids and Networks*, *6*, 81–90. doi: 10.1016/j.segan.2016.02.002

Crawley, D. B., Pedersen, C. O., Lawrie, L. K., & Winkelmann, F. C. (2000). Energyplus: Energy simulation program. *ASHRAE Journal*, *42*, 49–56.

Dulac-Arnold, G., Mankowitz, D. J., & Hester, T. (2019). Challenges of real-world reinforcement learning. *CoRR*, *abs/1904.12901*. Retrieved from `http://arxiv.org/abs/1904.12901`

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., & Madry, A. (2019). Implementation matters in deep rl: A case study on ppo and trpo. In *International conference on learning representations.*

Gajane, P., Ortner, R., & Auer, P. (2019, may). Variational Regret Bounds for Reinforcement Learning. *35th Conference on Uncertainty in Artificial Intelligence, UAI 2019*. Retrieved from `http://arxiv.org/abs/1905.05857`

Hallak, A., Di Castro, D., & Mannor, S. (2015, feb). Contextual Markov Decision Processes. *arXiv preprint arXiv:1502.02259*. Retrieved from `http://arxiv.org/abs/1502.02259`

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hosseinloo, A. H., Ryzhov, A., Bischi, A., Ouerdane, H., Turitsyn, K., & Dahleh, M. A. (2020, jan). Data-driven control of micro-climate in buildings; an event-triggered reinforcement learning approach. *arXiv preprint arXiv:2001.10505*. Retrieved from `http://arxiv.org/abs/2001.10505`

Iyengar, G. N. (2005, may). *Robust dynamic programming* (Vol. 30) (No. 2). INFORMS. doi: 10.1287/moor.1040.0129

Jaksch, T., Ortner, R., & Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, *11*(Apr), 1563–1600.

Kakade, & Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *Icml* (Vol. 2, pp. 267–274).

Kakade, S. M. (2002). A natural policy gradient. In *Advances in neural information processing systems* (pp. 1531–1538).

Kim, D. W., & Park, C. S. (2011, dec). Difficulties and limitations in performance simulation of a double skin façade with EnergyPlus. *Energy and Buildings*, *43*(12), 3635–3645. doi: 10.1016/j.enbuild.2011.09.038

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... Hadsell, R. (2017, mar). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, *114*(13), 3521–3526. doi: 10.1073/pnas.1611835114

Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in neural information processing systems* (pp. 1008–1014).

Kong, W., Dong, Z. Y., Jia, Y., Hill, D. J., Xu, Y., & Zhang, Y. (2019). Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. *IEEE Transactions on Smart Grid*, *10*(1), 841–851.

Lecarpentier, E., & Rachelson, E. (2019). Non-Stationary Markov Decision Processes a Worst-Case Approach using Model-Based Reinforcement Learning. In *Ad-*

vances in neural information processing systems (pp. 7214–7223).

Li, Y., Wen, Y., Tao, D., & Guan, K. (2019, jul). Transforming Cooling Optimization for Green Data Center via Deep Reinforcement Learning. *IEEE Transactions on Cybernetics*, *50*(5), 2002–2013. doi: 10.1109/tcyb.2019.2927410

Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J. E., & Stoica, I. (2018). Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., … Wierstra, D. (2016, sep). Continuous control with deep reinforcement learning. In *4th international conference on learning representations, iclr 2016 - conference track proceedings.* International Conference on Learning Representations, ICLR.

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

Mankowitz, D. J., Mann, T. A., Bacon, P.-L., Precup, D., & Mannor, S. (2018, apr). *Learning Robust Options* (Tech. Rep.). Retrieved from `www.aaai.org`

Mao, J., & Jain, A. K. (1995). Artificial neural networks for feature extraction and multivariate data projection. *IEEE transactions on neural networks*, *6*(2), 296–317.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015, feb). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. doi: 10.1038/nature14236

Mocanu, E., Mocanu, D. C., Nguyen, P. H., Liotta, A., Webber, M. E., Gibescu, M., & Slootweg, J. G. (2018). On-line building energy optimization using deep reinforcement learning. *IEEE transactions on smart grid*, *10*(4), 3698–3708.

Mohajerin, N., & Waslander, S. L. (2019). Multistep prediction of dynamic systems with recurrent neural networks. *IEEE transactions on neural networks and learning systems*, *30*(11), 3370–3383.

Moriyama, T., De Magistris, G., Tatsubori, M., Pham, T. H., Munawar, A., & Tachibana, R. (2018, oct). Reinforcement Learning Testbed for Power-Consumption Optimization. In *Communications in computer and information science* (Vol. 946, pp. 45–59). Springer Verlag.

Nagy, A., Kazmi, H., Cheaib, F., & Driesen, J. (2018, may). Deep Reinforcement Learning for Optimal Control of Space Heating. *arXiv preprint arXiv:1805.03777*. Retrieved from `http://arxiv.org/abs/1805.03777`

Naug, A., Ahmed, I., & Biswas, G. (2019, jun). Online energy management in commercial buildings using deep reinforcement learning. In *Proceedings - 2019 ieee international conference on smart computing, smartcomp 2019* (pp. 249–257). Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/SMARTCOMP.2019.00060

Naug, A., & Biswas, G. (2018). Data driven methods for energ reduction in large buildings. In *2018 ieee international conference on smart computing (smartcomp)* (pp. 131–138).

Padakandla, S., J., P. K., & Bhatnagar, S. (2019). Reinforcement learning in non-stationary environments. *CoRR*, *abs/1905.03970*. Retrieved from `http://arxiv.org/abs/1905.03970`

Park, C.-S. (2013, 08). Difficulties and issues in simulation of a high-rise office building..

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, Inc.

Rahman, A., Srikumar, V., & Smith, A. D. (2018, feb). Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks. *Applied Energy*, *212*, 372–385.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015, feb). Trust Region Policy Optimization. *32nd International Conference on Machine Learning, ICML 2015*, *3*, 1889–1897. Retrieved from `http://arxiv.org/abs/1502.05477`

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shaikh, P. H., Nor, N. B. M., Nallagownden, P., Elamvazuthi, I., & Ibrahim, T. (2014). A review on optimized control systems for building energy and comfort management of smart sustainable buildings. *Renewable and Sustainable Energy Reviews*, *34*, 409–429.

Shashua, S. D.-C., & Mannor, S. (2017, mar). Deep Robust Kalman Filter. *arXiv preprint arXiv:1703.02310*. Retrieved from `http://arxiv.org/abs/1703.02310`

Singh, N., Dayama, P., & Pandit, V. (2019). Change point detection for compositional multivariate data. *arXiv preprint arXiv:1901.04935*.

Smarra, F., Jain, A., De Rubeis, T., Ambrosini, D., D'Innocenzo, A., & Mangharam, R. (2018). Data-driven model predictive control using random forests for building energy optimization and climate control. *Applied energy*, *226*, 1252–1272.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 2951–2959). Curran Associates, Inc. Retrieved from `http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf`

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems* (pp. 1057–1063).

Tamar, A., Mannor, S., & Xu, H. (2014). Scaling up robust mdps using function approximation. In *International conference on machine learning* (pp. 181–189).

Wei, T., Wang, Y., & Zhu, Q. (2017). Deep reinforcement learning for building hvac control. In *Proceedings of the 54th annual design automation conference 2017* (pp. 1–6).

Zheng, X., Zaheer, M., Ahmed, A., Wang, Y., Xing, E. P., & Smola, A. J. (2017). State space lstm models with particle mcmc inference. *arXiv preprint arXiv:1711.11179*.