Generalized Reinforcement Learning for Building Control using Behavioral Cloning

Zachary E. Lee, K. Max Zhang

Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, USA

Abstract

Advanced building control methods such as model predictive control (MPC) offer significant benefits to both consumers and grid operators, but high computational requirements have acted as barriers to more widespread adoption. Local control computation requires installation of expensive computational hardware, while cloud computing introduces data security and privacy concerns. In this paper, we drastically reduce the local computational requirements of advanced building control through a reinforcement learning (RL)-based approach called Behavioral Cloning, which represents the MPC policy as a neural network that can be locally implemented and quickly computed on a low-cost programmable logic controller. While previous RL and approximate MPC methods must be specifically trained for each building, our key improvement is that the proposed controller can generalize to many buildings, electricity rates, and thermostat setpoint schedules without additional, effort-intensive retraining. To provide this versatility, we have adapted the traditional Behavioral Cloning approach through two innovations: (1) a constraint-informed parameter grouping (CIPG) method that provides a more efficient representation of the training data and (2) a new deep learning model-structure called reverse-time recurrent neural networks (RT-RNN) that allows future information to flow backward in time to more effectively interpret the temporal information in disturbance predictions. The result is an easy-to-deploy, generalized behavioral clone of MPC that can be implemented on a programmable logic controller and requires little building-specific controller tuning, reducing the effort and costs associated with implementing smart residential heat pump control.

Keywords: Deep Reinforcement Learning; Behavioral Cloning; Model Predictive Control; Smart Grid; Heat Pump;

Highlights

- Behavioral Cloning reduces model predictive control (MPC) computational requirements.
- One Behavioral Cloning agent generalizes to new buildings without further training.
- Constraint-informed parameter groupings provide more efficient state representations.
- Reverse-Time Recurrent Neural Networks incorporate future disturbance predictions.
- Simulations show Behavioral Cloning offers energy efficient control similar to MPC.

*Corresponding author: kz33@cornell.edu

1. Introduction

As the energy system relies more and more on variable renewable energy sources, efficient grid-interactive buildings that can modulate their demand according to the availability of renewable energy become ever more important. Buildings are becoming increasingly electrified, replacing fossil fuel based space heating with clean, electric alternatives such as heat pumps. A substantial amount of research has shown that smart heat pump control can harness the inherent thermal storage of the building envelope and provide important grid services such as load shifting and demand response [1], which are generally considered as requirements for maintaining a reliable electrical grid with high penetrations of renewable energy resources [2]. However, more advanced control methods that can provide this demand flexibility still face large technical, economical, and social barriers to adoption, and therefore they lack the scalability needed to have a large impact on the overall energy system.

One of the most widely studied advanced building control methods is model predictive control (MPC) [3]. Compared to conventional rule-based approaches, MPC can offer substantial energy consumption savings of 20% or more [4], as well as achieve other control objectives such as peak load reduction [5] and demand response [6]. At each time step, a constrained optimization problem is solved to determine the optimal control given a model of the building and predictions of future disturbances like weather, occupancy, and electricity prices. But despite substantial research efforts into the development of MPC for heat pumps, it has yet to be widely adopted due to its costly installation and computational hardware costs [7]. With over 30% of US households already reporting some difficulty in paying their energy bills [8], these high capital costs can make advanced building control economically unfeasible for low-income populations and neglect a significant source of demand flexibility.

One potential avenue for more scalable building control is through smart thermostats, which feature a simple plug-and-play installation that has resulted in rapid recent adoption [9]. However, smart thermostats have limited computational hardware that often cannot handle the high memory and processing requirements needed to solve MPC. Instead, smart thermostats use rule-based approaches for energy efficiency and demand response, but can connect to the cloud for more advanced data processing. While they can reduce energy consumption, these rule-based control methods often provide insufficient perceived benefit to justify the high capital costs of smart thermostats. In a recent US nationwide survey, 30% of people said that smart thermostats are too expensive and 60% said that they simply do not see the merits of upgrading their current system [10]. Moreover, for cloud-based smart thermostats, data privacy and security are other key concerns [11]. Thus, inexpensive plug-and-play control solutions that provide higher cost savings while preserving data privacy can reduce many of the barriers to more widespread adoption of smart heat pump control.

Some studies have simplified the MPC computation by analytically deriving its closed-form solution, called explicit MPC, so that it can be computed locally on low-cost, resource-constrained devices such as programmable logic controllers (PLC), which typically have limited processing speeds

in the range of MHz and memory on the order of hundreds of kB. Explicit stochastic MPC [12] and explicit scenario-based MPC [13] have both been shown to drastically improve MPC computational efficiency for building climate control with relatively small state spaces and time horizons. However, as the problem size gets larger, explicit MPC requires substantially more memory, meaning that the longer time horizons needed to achieve the benefits of building MPC can be prohibitively large. In addition, the important mixed-integer constraints like minimum cycle times or minimum compressor speeds [14] make these closed-form solutions to the optimal control problem significantly more difficult to derive.

More recently, machine learning-based control approaches have emerged as powerful tools for learning optimal control policies for systems with large state spaces and time horizons. Rather than analytically deriving the mapping from the state to the optimal control, these approaches leverages techniques like decision trees and deep learning to learn an approximate mapping using data collected from the system. One approach called Approximate MPC (AMPC) was recently proposed by Drgonaňa et al. [15] that approximated the explicit MPC formulation for a single building by training a feed-forward neural network on samples from closed-loop MPC simulations generated in EnergyPlus [16]. AMPC was also successfully implemented experimentally for an office building in Ref. [17]. Other reinforcement learning (RL) approaches like Deep Q-learning [18] and Asynchronous Advantage Actor Critic (A3C) [19] have been used to learn optimal control policies from scratch by interacting with a virtual building simulated in software like EnergyPlus.

However, the main problem with these approaches is that data generation is very time and labor intensive. A new controller must be trained for each specific building installation, and can require months to years of data samples to learn that building's optimal control policy. While large commercial buildings can cover the high capital costs required to develop virtual buildings to generate this data, small buildings and residences cannot. As a result, the fact that these machine learning-based building controllers cannot generalize to different buildings without expensive model retraining has been noted as a key obstacle to implementation [20].

In this paper, we combine ideas from both AMPC and RL research to create a resource efficient optimal controller that can generalize to many buildings while being trained only once, significantly reducing the capital costs and installation effort to implement more advanced building control. In particular, we apply a form of reinforcement learning called *Behavioral Cloning* [21] that has to our knowledge not been applied in the building control context. Behavioral Cloning attempts to mimic the actions of an available expert controller, such as MPC, and results in much faster training compared to other RL approaches [22]. By training one controller on a large number of simulated buildings, setpoint schedules, and electricity rates up front, the controller can generalize to different buildings, various resident preferences, and changing utility prices without additional controller tuning.

In addition, we improve the conventional Behavioral Cloning approach to make it more suitable for building control. First, we introduce a more efficient representation of the input state using MPC

constraint-informed parameter groupings (CIPG). Second, we present a new machine learning model structure called reverse-time recurrent neural networks (RT-RNN) that more accurately interprets the future disturbance information that is vital to effectively pre-heat or pre-cool the building for energy efficiency. The result is a behavioral clone of MPC that uses only around 100 kB of memory, requires negligible computational time, and can be implemented in buildings on a PLC in a low-cost thermostat with minimal installation effort and costs.

We begin the paper with an overview of Behavioral Cloning and related work in Section 2. Section 3 then provides our proposed methodology. Next in Section 4, we implement the methodology on a common heat pump system architecture. Section 5 tests this implementation on a population of simulated buildings and operating conditions and compares the results and computational requirements to MPC and a baseline rule-based control. Section 6 concludes the paper.

2. Preliminaries

2.1. Behavioral Cloning

While most RL applications require a very large amount of training data to learn the optimal control policy from scratch, Behavioral Cloning can learn the policy much more efficiently by taking advantage of an available expert controller [23]. Behavioral Cloning is typically used when training data generation is expensive or time consuming and when the expert controller is impractical to deploy. For example, the most common Behavioral Cloning application has been to mimic human drivers for autonomous driving purposes [24]. Another example which uses MPC as the expert controller was given in [25] to control a walking robot with faster online control. Behavioral Cloning can even learn policies only from observations by inferring the actions of the expert policy [26].

Behavioral Cloning seeks to learn a stochastic policy $\hat{\mu}$ that provides probabilities $\hat{\mu}(u|s)$ of taking a control u that most closely matches the expert control policy $\mu^*(s)$. Here, the state s contains all of the information relevant to solving the optimal problem, such as current and past measurements and future disturbances. Finding the approximate, or learner, policy $\hat{\mu}$ is a supervised learning problem that seeks to minimize the difference between the predicted control and the expert control given the input state. A diagram of the behavioral cloning process is given in Fig. 1.

This initial description of Behavioral Cloning is virtually the same as AMPC, which also learns an approximate optimal policy from samples generated by an expert MPC control. However, the key difference is how the training samples are generated: In AMPC, they are generated through closed-loop MPC simulations. For some applications such as in [15] and [17], where the operating conditions do not vary much, the control prediction can make very few errors and can provide statistical guarantees on constraint satisfaction and stability [27]. However, if the agent faces new operating conditions, such as changing setpoints or different electricity prices, the agent will likely make mistakes and deviate from the optimal control trajectory. Since closed-loop MPC simulations have no information outside of the optimal control trajectory, they are unable to provide sufficient

information for the agent to correct itself should it drift to a sub-optimal state, often known as the compounding error problem.

2.2. DAgger Algorithm

Behavioral cloning solves the compounding error problem by generating additional data outside of the optimal control trajectory. One of the most common methods for generating this data is called Dataset Aggregation (DAgger) [28]. DAgger is used in many different Behavioral Cloning applications, ranging from natural language processing [29] to autonomous driving [22]. DAgger is an iterative algorithm that uses the Behavioral Cloning agent to control the system, while the expert controller records the correct control decision at each time step to teach the agent how to recover from mistakes. DAgger therefore enriches the training dataset above pure closed-loop MPC simulations to allow the agent to be stable on new operating conditions and correct for control policy imperfection.

The general DAgger algorithm is as follows. At iteration $i = 0$, initialize the policy by simulating an episode controlled by the expert policy and train the Behavioral Cloning agent on the resulting state-control samples $(s, u^\star)$. For each following iteration, simulate another episode this time controlled by the Behavioral Cloning agent. At first, the agent will likely perform poorly and deviate from the optimal control trajectory due to the limited data. However, at each time step, the expert controller calculates and records, but does not implement, the true optimal control. At the end of each iteration, the optimal control solutions are added to the training data set and the agent is retrained with the additional data. Through this process, the correct control responses to suboptimal states are added to the training dataset so the agent can know how to correct itself in the future. These iterations can be repeated until the agent is stable during the testing phase and its objective value is within some limit of the true MPC objective value.

3. Methodology

Our methodology for Behavioral Cloning of building MPC is outlined in Fig. 2. To improve the generalization ability to multiple buildings and operating conditions, our methodology contains several tools to encode domain knowledge that increase training efficiency. The following section describes the general building control system and our contributions.

3.1. General Building Control System

While in practice building control problems can take many forms, they can often be reduced to three general components: (1) the heating or cooling sources, (2) the storage medium, and (3) the cost signal [30], illustrated in Fig. 2. By making very few assumptions on the structure of the control problem, we design our methodology to be applicable to a wide a range of building system architectures which require only minor changes to the MPC formulation and input state. For each

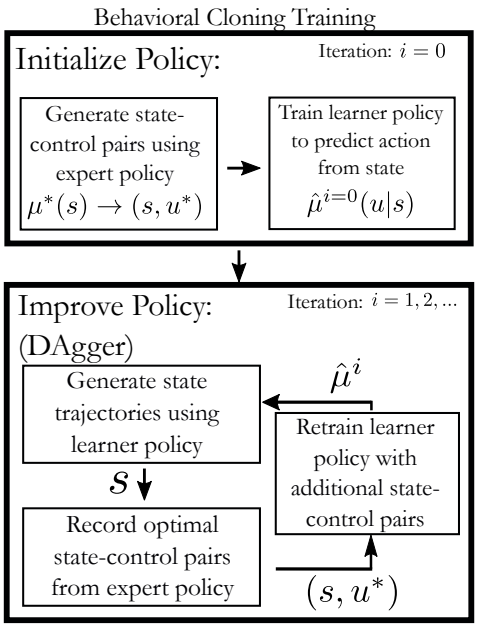


Figure 1: Behavioral Cloning training process. After initializing the learner policy using the expert controller, the learner policy then is used to control the system in order to generate samples outside of the optimal control trajectory, improve the policy, and solve the compounding error problem.

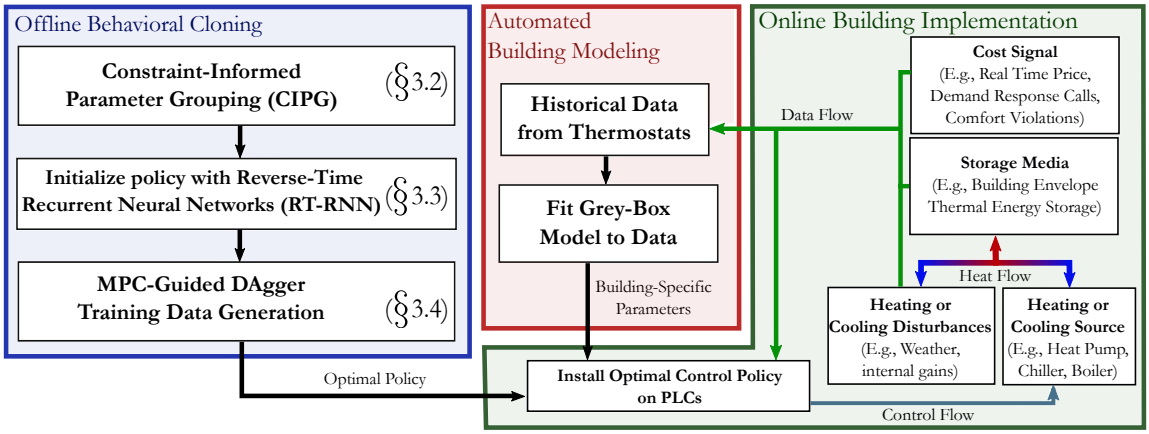


Figure 2: Overall methodology for Behavioral Cloning of MPC and implementing it in a population of buildings. Behavioral Cloning and building modeling can both be performed offline using a desktop computer, while online control only requires a programmable logic controller (PLC). The Online Building Implementation section shows a typical building control system structure.

class of building system architectures, only one controller must be trained for the buildings in that class. The primary assumption we make is to apply our methodology to relatively small buildings that can be captured using reduced-order models (see Sec. 4.1 for further discussion), since larger commercial buildings often have the ability to afford more advanced control hardware and develop virtual buildings.

In essence, the heating and cooling equipment only act as heating or cooling sources that add or remove heat from the storage medium in order to maintain indoor thermal comfort specified by the thermostat. The exact thermodynamic mechanism, however, can be very different. For example, heat pumps can be single stage (binary control) or variable speed (continuous control) and can exchange heat with either the outdoor air or the ground. While these differences strongly affect how strongly the efficiency varies (e.g. air-source heat pumps are much more dependent on the outdoor air temperature), the effects on efficiency follow the same general model and can be encoded into the state representation. In addition, weather and internal gains also act as heating or cooling sources, but since they are not controllable, we refer to them as disturbances.

The storage medium most often takes the form of the building's thermal mass, meaning that the storage limits are subject to the user-defined thermal comfort preferences. The limits on the storage medium define how much the heat pump can shift its operation toward times of higher efficiency or lower cost. In some cases, additional thermal storage like water tanks or phase change material can also be added, increasing the storage potential and reducing its dependency on user preferences.

Finally, the cost signal defines the desired grid service to be provided by the demand flexibility. The most basic embodiment of the cost signal is a time-of-use or real time price, which utilities use to encourage load shifting. However, other grid services like demand response or flexible ramping [31] can be encoded into the cost signal using artificially high costs during a period when the utility needs to reduce demand. By imposing penalties for violating thermal comfort [32], the optimal controller balances the resident's desired tradeoff between saving money and maintaining thermal comfort.

3.2. Constraint Informed Parameter Groupings (CIPG)

In order to provide a more sample efficient and generalizable representation of the state and disturbance inputs, we developed a method called *constraint-informed parameter groupings (CIPG)*, which group parameters based on the structure of the MPC constraints and building model. This approach is inspired by dimensionality reduction ideas from of the Buckingham Pi Theorem. As an example, this method is used in fluid dynamics to non-dimensionalize fluid parameters such that the solutions to complex fluid flows are no longer functions of the actual parameter values (e.g., viscosity, velocity, temperature, etc.), but instead functions of the ratios between the values (e.g., Reynolds number). Similarly, the building thermodynamics and the optimal control are not necessarily functions of the actual parameter values, but rather the ratios or differences between the parameter values (e.g., heat loss is a function of the temperature difference). Therefore, by grouping

the training data parameters based on the constraints in the MPC formulation, we condense the feature space to allow operating conditions from one training simulation to be more effectively applied to a different operating condition during test time.

The parameter groupings can be loosely classified into four classes. The first class of parameter groupings represents the building and heat pump model, which is what allows the agent to generalize across buildings. Rather than using a black-box or white-box model, which can sometimes require hundreds of unique parameters, we use a reduced-order grey-box building model to allow the thermodynamics to be efficiently grouped as part of the state information. The second class includes the external effect of weather on the building. The third provides the limits of the storage medium and thermal comfort. The fourth describes the cost signal. A derivation of the parameter groupings for a specific control problem can be found in Sec. 4.3.

3.3. Control Policy Parameterization

The type of machine learning model structure is especially important to develop a functioning Behavioral Cloning agent. Particularly with MPC, the ability to extract the temporal information embedded in the disturbance forecasts heavily affects the model's performance. Knowing that the setpoint will rise at a specific time in the future determines at what time the agent should begin preheating. In most RL applications such as [33], these future disturbances are implicitly predicted by the control policy using a state representation with a sufficient number of previous timesteps. However, future disturbances such as weather are independent of the policy, and can be better predicted separately using available weather forecasts *without* the need for expensive building simulations to generate this data. Conventional supervised learning techniques previously used in approximate MPC [15] like regression trees and feed-forward neural networks do not contain any inherent structure to interpret this future temporal information and thus were not sufficient to learn the larger feature space and be able to generalize to new conditions. Therefore, we propose a new model structure called reverse-time recurrent neural networks (RT-RNN) to better capture the temporal information contained in the future disturbance predictions.

Traditional recurrent neural networks (RNN) are a type of neural network that use a time-based structure to take advantage of temporal information in the data. RNNs take inputs from the current time step and from previous time steps that are passed through the RNN layer as a hidden state. RNNs perform significantly better than conventional feed-forward neural networks (FFNN) on sequential data applications such as forecasting and natural language processing [34].

In our case, however, the input features do not contain data from previous time steps, but rather from future disturbances like weather, electricity price, and setpoint preferences. Nevertheless, future disturbances can also benefit from being used in RNNs, as is in the case of bidirectional RNNs, which use both previous and future datapoints to make a prediction at the current time step [35]. We apply this idea to Behavioral Cloning in the form of reverse-time RNNs, where the RNN is structured such that time is reversed, and future disturbance prediction information flows

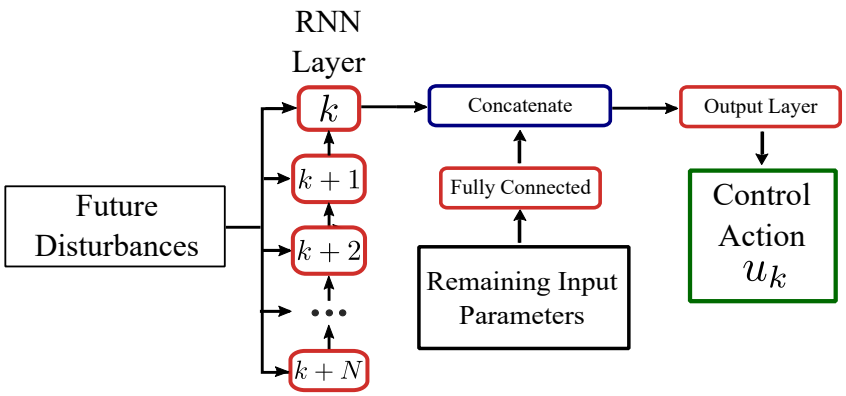


Figure 3: Reverse-Time Recurrent Neural Network Structure. Future disturbance parameter groupings containing weather, electricity price, and setpoint information are input into the reverse RNN layer where information flow backward in time over MPC horizon. These are concatenated with inputs from the other parameters and then to the output layer.

backward in time to help predict the optimal control at the current time step.

Since RNNs contain feedback loops to store memory, they can experience vanishing or exploding gradients if the sequences are too long. Thus, vanilla RNNs are often unable to learn long term temporal dependencies. To solve this, RNNs have been improved with model structures like gated recurrent units (GRU) [36] and long short-term memory (LSTM) [37], which are capable of storing a separate memory state that may be important in a long sequence. These structures can be equally applied for RT-RNNs, where the *memory state* can instead be termed the *prediction state*. For example, if a setpoint change occurs several hours in the future, the prediction state can store this information without it being potentially lost due to vanishing gradients over many time steps in the RNN. While LSTMs often outperform GRUs due to a more complex structure, GRUs can be more suited for memory constrained applications or on smaller datasets [38]. Therefore, in Sec. 4.4.1, we test our model structure using both layer types against three other supervised learning methods to show that RT-RNNs can provide the best performance while maintaining minimal memory and processing requirements.

Our proposed RT-RNN structure is given in Fig. 3. The time-dependent parameter groupings that contain future disturbance information are input into the RNN layer. The information then flows backward in time, from the end of the MPC horizon to the current time step. The output of this layer is concatenated with the output of the remaining input parameters put through a fully connected layer. The final output layer contains a sigmoid activation function to give the binary control action prediction. In the case of continuous control, a linear activation function can also be used.

3.4. Model Training and Implementation

To train the Behavioral Cloning agent, we first generate a large amount of training data using the DAgger algorithm. For each episode, new building and heat pump model parameters are randomly generated and new operating conditions are selected from a large source of weather, cost, and setpoint data. Our specific implementation of the DAgger algorithm is given by Algorithm 1.

Algorithm 1: MPC-Guided DAgger

Simulate a set of randomized buildings, weather, and electricity tariffs using MPC;
Train Behavioral Cloning agent on resulting normalized dataset;
while $J_{appr} - J_{MPC} \leq \varepsilon$ **do**
  Simulate new set of randomized buildings, weather, and electricity tariffs controlled using the agent;
  At each time step, solve MPC and add (but do not implement) the inputs and solutions to training dataset;
  Retrain agent with additional training data;
  Evaluate agent on test conditions and calculate total objective value J_{appr};
end

After the data is generated, we optimize the policy parameterization by tuning hyperparameters to achieve the highest control prediction accuracy while minimizing the model's required memory consumption. This training is done offline and must only be done once per class of building system architectures.

Next, the optimal model is implemented in a test simulation on a sample of buildings intended to mimic real-world operation. To implement the controller, a homeowner buys and installs a low-cost thermostat containing a PLC with the Behavioral Cloning agent installed. The thermostat then collects various operational data over a period of time that can be used to automatically derive a data-driven reduced order building model using the method given in [5]. These model parameters, combined with weather forecasts and data collected by the thermostat, are then used as inputs to the Behavioral Cloning agent to provide online approximately optimal control. This test simulation contains buildings with diverse thermodynamics and heat pump performances, various thermostat setpoint schedules obtained from real data, and different electricity price schedules, all of which were not originally included in the training dataset. By testing on these diverse operating conditions, we show our Behavioral Cloning approach leads to improved versatility and minimal-effort implementation compared to the current state-of-the-art building AMPC [15].

4. Case Study Formulation

While our methodology is designed to be applicable to a wide range of residential building types and heat pump configurations, we test our methodology on one of the most common residential

configurations: a detached home served by a single-stage air-to-air heat pump. However, many heat pump MPC formulations consist of similar structures, and thus it is straightforward to adapt our methodology to other system types for both heating and cooling.

4.1. Model Definition

To be able to model each system without significant manual effort, we use a data-driven grey-box model, where each of the building and heat pump model parameters can be automatically identified from collected data. While deriving these building parameters sometimes requires data collected from a variety of sensors throughout the building [39] or from building energy simulations [40], we use the identification and control method presented in [5], which is designed to require minimal hardware installation cost and effort.

The building model can be represented by a thermal resistance-capacitance (RC) circuit. RC models are widely used in the building control literature and are applicable to a wide range of buildings [30]. In addition, multi-state models can capture the increased energy storage capacity of the building's construction [41]. We use a two-state model that includes different states for the building's indoor air and the building's construction and includes effects from solar irradiation, given by [5],

$$\begin{aligned} C_a\dot{T}_a(t) &= \frac{T_\infty - T_a(t)}{R_{a\infty}} + \frac{T_m(t) - T_a(t)}{R_{am}} + \alpha_a G + Q_{\text{HP}} \\ C_m\dot{T}_m(t) &= \frac{T_\infty - T_m(t)}{R_{m\infty}} + \frac{T_a(t) - T_m(t)}{R_{am}} + \alpha_m G, \end{aligned} \tag{1}$$

where the subscript a refers to the indoor air, m to the building mass, and ∞ to the outside air. The resistance and capacitance values are given by R and C, respectively, while the temperature of the states is given by T. Solar heat gains are included using the solar irradiation G and the solar absorption factor α. Based on an analysis of manufacturer performance data [42], the heat transfer from the heat pump Q_{HP} are assumed to vary linearly based on the indoor and outdoor temperature, given by,

$$Q_{\text{HP},a} = u(\beta_1(T_\infty - T_a) + \beta_2). \tag{2}$$

where β_i are data-driven heat pump specific model parameters and u denotes the binary control input for whether the heat pump is on or off.

Finally, based on an analysis of data in [42] the power consumption P is assumed to be a constant γ multiplied by the control input,

$$P = \gamma u$$

For use in MPC, the model is discretized with time step Δt into the state space form indexed by k,

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B_k\boldsymbol{u}_k + E\boldsymbol{w}_k, \tag{3}$$

where,

$$\boldsymbol{x} = \begin{bmatrix} T_{a,k} \\ T_{m,k} \end{bmatrix}, \quad \boldsymbol{u_k} = \begin{bmatrix} u_k \end{bmatrix}, \quad \boldsymbol{w_k} = \begin{bmatrix} T_{\infty,k} \\ G_k \end{bmatrix}$$

$$A = \begin{bmatrix} 1 - \frac{\Delta t}{C_a}\left(\frac{1}{R_{a\infty}} + \frac{1}{R_{am}}\right) & \frac{\Delta t}{C_a R_{am}} \\ \frac{\Delta t}{C_m R_{am}} & 1 - \frac{\Delta t}{C_m}\left(\frac{1}{R_{m\infty}} + \frac{1}{R_{am}}\right) \end{bmatrix},$$

$$B_k = \begin{bmatrix} \frac{\Delta t}{C_a}\left(\beta_1(T_{\infty,k} - T_{\text{set},k}) + \beta_2\right) \\ 0 \end{bmatrix},$$

$$E = \begin{bmatrix} \frac{\Delta t}{R_{a\infty} C_a} & \frac{\alpha_a \Delta t}{C_a} \\ \frac{\Delta t}{R_{m\infty} C_m} & \frac{\alpha_m \Delta t}{C_m} \end{bmatrix}.$$

4.2. Control Formulation

The controller seeks to minimize the time-varying electricity cost while maintaining thermal comfort in response to varying thermostat setpoints decided by the resident. We define thermal comfort as a temperature range above and below the thermostat setpoint. Since we assume that setpoints are customizable by the resident, to maintain feasibility we penalize violations outside of this thermal comfort band. These violations are enforced by the constraints,

$$\begin{aligned} T_{k+j} &\leq T_{\text{set},k+j} + T_{\delta,k+j} + \overline{T}_{\text{pen},k+j} \quad \forall j \in N \\ T_{k+j} &\geq T_{\text{set},k+j} - T^i_{\delta,k+j} - \underline{T}_{\text{pen},k+j} \quad \forall j \in N. \end{aligned} \tag{4}$$

Here, $T_{pen,k+j}$ is the comfort violation decision variable, $T_{\delta,k+j}$ is the resident's specified comfort band above or below the setpoint, $T_{\text{set},k+j}$ is the resident's specified setpoint, and N represents the prediction horizon indexed by j. Note that the comfort band can also vary based on time of day and can be determined by whether the thermostat is in home, away, or sleep modes.

Next, heat pumps have inherent minimum on and off times to prevent short cycling and the resulting compressor damage and efficiency reduction. To enforce these minimum cycle times, we add the following constraints,

$$u_{k+j} - u_{k+j-1} = v^{\uparrow}_{k+j} - v^{\downarrow}_{k+j} \quad \forall j \in N \tag{5}$$

$$\sum_{i=k+j-t_{\text{min on}}}^{k+j} v^{\uparrow}_i \leq u_{k+j} \quad \forall j \in N \tag{6}$$

$$\sum_{i=k+j-t_{\text{min off}}}^{k+j} v^{\downarrow}_i \leq 1 - u_{k+j} \quad \forall j \in N. \tag{7}$$

Here, $v^{\uparrow}_i$ and $v^{\downarrow}_i$ are binary variables that are unity if the heat pump turned on or off, respectively, at the time step i. The parameters $t_{\text{min, on}}$ and $t_{\text{min, off}}$ are the minimum on and off times, respectively.

The objective function combines the time-varying cost of electricity $\pi_{e.j}$ with the upper and lower thermal comfort penalties, $\underline{\pi}_{\text{pen}}$ and $\overline{\pi}_{\text{pen}}$,

$$\min_{u_{k+j}} \quad J = \sum_{j=0}^{N-1} \left[\pi_{e,k+j} P_{k+j} + \underline{\pi}_{\text{pen}} \underline{T}_{\text{pen},k+j} + \overline{\pi}_{\text{pen}} \overline{T}_{\text{pen},k+j} \right] \tag{8}$$

The final MPC problem is therefore,

$$\min_{u_{k+j}} \quad \sum_{j=0}^{N-1} \left[\pi_{e,k+j} P_{k+j} + \underline{\pi}_{\text{pen}} \underline{T}_{\text{pen},k+j} + \overline{\pi}_{\text{pen}} \overline{T}_{\text{pen},k+j} \right] \tag{9a}$$

subject to

$$\boldsymbol{x}_{k+j+1} = A\boldsymbol{x}_{k+j} + B_{k+j}\boldsymbol{u}_{k+j} + E\boldsymbol{w}_{k+j} \quad \forall j \in N \tag{9b}$$

$$T_{a,k+j} \leq T_{\text{set},k+j} + T_{\delta,k+j} + \overline{T}_{\text{pen},k+j} \quad \forall j \in N \tag{9c}$$

$$T_{a,k+j} \geq T_{\text{set},k+j} - T^{i}_{\delta,k+j} - \underline{T}_{\text{pen},k+j} \quad \forall j \in N \tag{9d}$$

$$u_{k+j} - u_{k+j-1} = v^{\uparrow}_{k+j} - v^{\downarrow}_{k+j} \quad \forall j \in N \tag{9e}$$

$$\sum_{i=k+j-t_{\text{min on}}}^{k+j} v_i^{\uparrow} \leq u_{k+j} \quad \forall j \in N \tag{9f}$$

$$\sum_{i=k+j-t_{\text{min off}}}^{k+j} v_i^{\downarrow} \leq 1 - u_{k+j} \quad \forall j \in N. \tag{9g}$$

$$\tag{9h}$$

This gives the optimal MPC policy $\mu^*_{\text{mpc}}(s)$ that maps the total state, which contains the building parameters and disturbance forecasts, to the optimal control u^*_k,

$$u^*_k = \mu^*(s) = \mu^*(\boldsymbol{x}_k, A, B_{k+j}, E, \gamma, T_{\text{set},k+j}, T_{\delta,k+j}, \pi_{e,k+j}, \underline{\pi}_{\text{pen}}, \overline{\pi}_{\text{pen}}).$$

where $\mu^*(s)$ is found numerically by solving the optimization problem.

4.3. Constraint Informed Parameter Groupings (CIPG)

In this section we derive the specific parameter groupings that reformulate the original state into the more generalizable state representation. Information to each of the following four classes should be characterized by at least one parameter grouping: (1) the building and heat pump model, (2) weather effects, (3) the storage medium and thermal comfort, and (4) the cost signal. For the first grouping, the building's thermodynamic parameters R, C are simply grouped as the entries of the A state space matrix defined in Eq. 3. Though this initial grouping is quite straightforward, it illustrates the point that it is not the parameter values themselves that govern the MPC solution, but the ratios of the parameters instead. Following the notation of the Buckingham Pi Theorem

where Π refers to a grouped parameter, the building model parameter groupings are given by the vector,

$$\Pi_1 = [a_{11}, a_{12}, a_{21}, a_{22}], \tag{10}$$

where the subscripts denote the entries in the corresponding state space matrix.

Next, the heat pump's effect on the indoor air temperature comes from the B matrix defined in Eq. 3. Since the heat output changes based on the indoor and outdoor air temperature, this parameter grouping is indexed by j over the MPC horizon N. The normalized parameter corresponding to the heat pump is given as,

$$\Pi_{2,k+j} = b_{11,k+j} \quad \forall j \in N. \tag{11}$$

The weather's effect on the solution comes from the forecasts for outdoor temperature and solar irradiation and the corresponding thermal properties of the home grouped in the C matrix defined in Eq. 3. We combine these into a matrix indexed over the MPC horizon,

$$\Pi_{3,k+j} = \begin{bmatrix} c_{11}T_{\infty,k+j} \\ c_{21}T_{\infty,k+j} \\ c_{12}G_{k+j} \\ c_{22}G_{k+j} \end{bmatrix}, \quad \forall j \in N. \tag{12}$$

We reformulate the thermal comfort constraints by taking the distance between the temperature at the current time step, $T_{a,0}$, and the upper and lower thermal comfort bounds indexed over the control horizon. Here, the upper and lower thermal comfort bounds represent the storage medium limits, and the current temperature represents the current storage state, and is defined such that value will be zero if $T_{a,0}$ is at the lower comfort bound and unity if it is at the upper comfort bound, given by,

$$\Pi_{4,k+j} = \frac{T_{a,k} - (T_{\text{set},k+j} - T_{\delta,k+j})}{2T_{\delta,k+j}} \quad \forall j \in N. \tag{13}$$

The normalized parameter corresponding to cost signal is the ratio between the electricity price at each time step over the control horizon and the average of the upper and lower thermal comfort penalty parameters, representing the tradeoff between cost savings and thermal comfort. It is then multiplied by γ to give the total energy cost of turning the heat pump on. This grouping is indexed over the MPC horizon and given by,

$$\Pi_{5,k+j} = \frac{2\gamma\pi_{e,k+j}}{\underline{T}_{\text{pen},k+j} + \underline{T}_{\text{pen},k+j}} \quad \forall j \in N. \tag{14}$$

Finally, we implement the minimum heat pump on and off time constraints by supplying the previous control values. Since we assume a 15-minute minimum heat pump cycle time and a five

minute time step, this becomes three previous control steps,

$$\Pi_{6,k} = [u_{k-1}, u_{k-2}, u_{k-3}] \tag{15}$$

The result is a new functional form for the MPC policy that is a function of the reformulated state (s_Π) that contains the parameter groupings and spans a reduced parameter space,

$$u^* = \mu^*(s_\Pi) = \mu^*(\Pi_1, \Pi_{2,k}, \Pi_{3,k}, \Pi_{4,k}, \Pi_{5,k}, \Pi_{6,k}). \tag{16}$$

4.4. Performance Evaluation

We evaluate the Behavioral Cloning agent in two steps: control prediction accuracy to determine the optimal Behavioral Cloning model structure and control simulation performance to determine its comparison to existing building control policies.

4.4.1. Policy Structure Optimization

We first evaluate control prediction accuracy to select the optimal policy structure and hyperparameter configuration. We compare the control prediction performance and computational requirements of the RT-RNN to three more conventional supervised learning techniques: (1) FFNNs, (2) Random Forest, and (3) Extreme Gradient Boosting (XGBoost). FFNNs represent the most basic deep neural network architecture, and pass information forward from the input features to the output prediction through multiple fully connected layers. Each node in a layer contains a vector of weights for each of the nodes in the previous layer and a bias parameter. The value of each node is then put through a nonlinear activation function to allow the network approximate nonlinear functions.

Random forest is an ensemble based supervised learning method that uses an ensemble of many different decision trees to classify data [43]. Different decision trees are fit based on random subsamples of the dataset, and each tree's output votes toward the final model's decision. By taking the majority vote of many decision trees, random forest reduces the potential for overfitting that is common with single decision trees. Both the memory requirement and performance of random forest depends on key hyperparameters that govern the number and size of the trees and must be optimized.

Extreme Gradient Boosting (XGBoost) is similar to random forest in that it uses an ensemble of decision trees, but it differs based on how the trees are created [44]. Instead of creating each tree independently, XGBoost uses extreme gradient boosting to iteratively improve a decision tree using more trees. At each iteration, the algorithm constructs a new tree to predict the error resulting from the previous ensemble of trees and then adds the new tree to the ensemble using a scaling factor called the learning rate. By doing so, the algorithm "boosts" the prediction at each step until no more performance gains can be made.

While more model parameters can theoretically learn more complex representations of the input data, this comes at the cost of larger model and higher memory requirements. To analyze this tradeoff, we determine each machine learning model's optimal hyperparameters through a grid search with 25 iterations for each model type. For each iteration, we log the model size and the validation prediction accuracy. Model size refers to the memory requirements to store each of the individual model parameters and is measured in kilobytes. Validation prediction accuracy refers to the model's prediction accuracy where the validation data is comprised of a random selection of 10% of the buildings simulated in the training data.

4.4.2. Control Simulation Performance

After selecting the best predicting model, the actual control performance is found through control simulations. We define control performance as the cumulative MPC objective function over a five-day test simulation on a set of buildings B, operating conditions, and electricity tariffs that were not included in the original training dataset, represented by the equation,

$$\sum_{b=0}^{B}\sum_{k=0}^{K}\left[\pi_{e,k}P_k^b + \underline{\pi}_{\text{pen}}\underline{T}_{\text{pen},k}^b + \overline{\pi}_{\text{pen}}\overline{T}_{\text{pen},k}^b\right] \tag{17}$$

Here k is the time step and K is the total number of time steps in the five-day test. Since setpoint preference and building thermal capacity can have a strong effect on MPC benefits, the model is tested on ten different buildings indexed by b to give a more holistic evaluation of model performance and generalization. Final computational requirements are logged during this simulation and include the processing speed and memory requirements required to store and run the model.

We use these metrics to compare the Behavioral Cloning control to a baseline standard rule-based control policy and the true MPC policy. In this case, the rule-based control policy is the typical thermostat's hysteresis control, where the heat pump turns on when the indoor temperature falls below the lower comfort bound and turns off when the temperature rises above the upper comfort bound. Note that this rule-based policy uses variable setpoint schedules that may include energy-saving setbacks when the occupant is away or asleep. In contrast, the MPC policy provides the target objective function value that Behavioral Cloning is trying to imitate.

5. Case Study Results

5.1. DAgger Training Data Generation

At each iteration of the DAgger algorithm, the system simulates new buildings with different random R, C, and α values and different heat pump performance coefficients. Various setpoint schedules were obtained from the Ecobee Donate Your Data dataset [45], which contains smart thermostat setpoint schedules from thousands of homes throughout the country. Thermal comfort band schedules were set based on whether those thermostats were in "home", "sleep", or "away" modes. We assume the comfort band is ± .5°C for "home", ± 1.0°C for "sleep", and no limit when

"away". Electricity price schedules were obtained from New York State Electric and Gas (NYSEG) [46], ConEdison (ConEd) [47], and Xcel Energy [48], three utilities that offer time-of-use rates during winter. Weather data comes from various days in January and February 2019 for New York City [49]. Note that while our method provides some level of generalization, if the climate varies significantly from training case, more simulations specific to the target climate may be required.

We generated 15 days of training data, each containing 10 randomized buildings, heat pumps, and setpoint schedules. For each of the random buildings, the thermodynamic model parameters were randomly selected from a range of $\pm 25\%$ around the values used in [5]. This totals to 45,760 samples of data used for training. To show the benefit of both the constraint-informed parameter normalization and the DAgger algorithm, we trained a set of models on three different training data representations. The first (CIPG + DAgger) is the aforementioned dataset generated by DAgger and normalized using our constraint-informed parameter groupings. The second (CIPG + AMPC) uses our constraint-informed parameter groupings, but instead is trained to approximate the closed loop MPC simulations and thus contains no information outside of the optimal control trajectory. The third (No Parameter Groupings + DAgger) contains data from the DAgger-generated dataset that is independently normalized. In other words, the third dataset uses only the conventional machine learning approach of scaling each individual input variable to have zero mean and unit variance, rather than our approach of first creating CIPGs and then scaling.

To compare the datasets, we trained 25 RT-RNNs for each dataset using various hyperparameter combinations to find the combination that provided the highest prediction accuracy on validation data. We then tested each dataset's best model in a control simulation containing new conditions outside of the training dataset. Control performances for each dataset are shown in Fig. 4.

There are two important findings from these results. First, combining the features into parameter groupings in CIPG + DAgger provides a three percentage point increase in validation prediction accuracy over No Parameter Groupings + DAgger, meaning that it has an improved ability to generalize outside of the training data distribution. While there is no significant difference in electricity cost, the improved prediction accuracy translates to significantly reduced comfort violations. Second, despite lower validation accuracy, Behavioral Cloning trained with DAgger has an order of magnitude better control performance than the AMPC model, which was trained to approximate closed loop MPC. The higher accuracy on the AMPC dataset is somewhat misleading and does not translate to better control performance. Since it is trained on closed loop MPC simulations the data is more homogeneous, and the indoor temperature is always within the thermal comfort limits. This contrasts with the DAgger dataset, which has data across a range of indoor temperatures, particularly from early iterations when the model does not perform well. The implication is that while it is easier to fit a more homogenous dataset, the AMPC model has insufficient data to correct itself if it strays from the optimal trajectory, and the result is a model with no knowledge that comfort violations are undesirable.

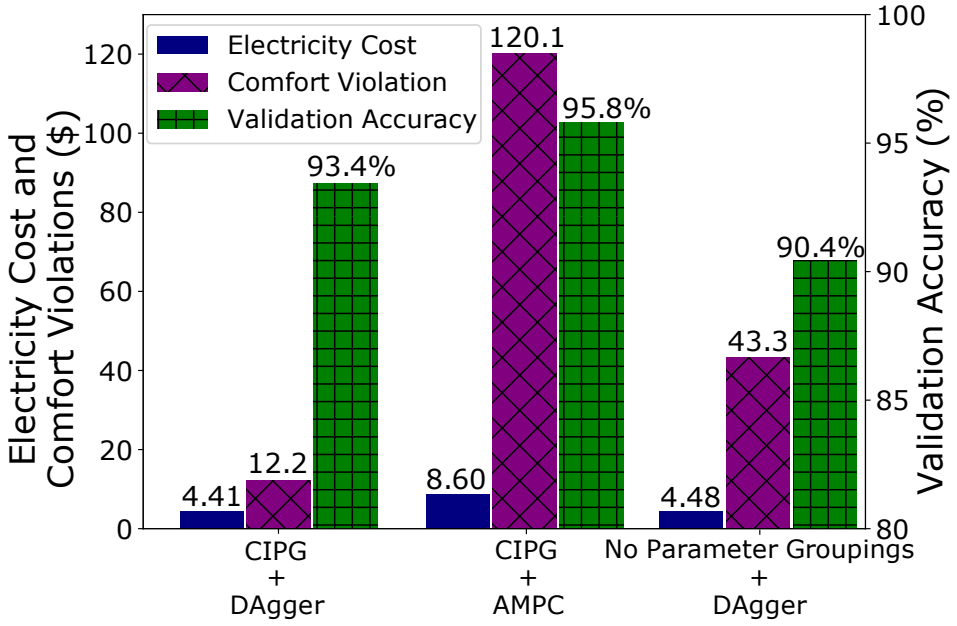


Figure 4: Per-unit Control Objectives (lower is better) and Validation Prediction Accuracy (higher is better) for (1) constraint informed parameter groupings (CIPG) with DAgger training data, (2) CIPG and trained to approximate closed loop MPC (AMPC), and (3) CIPG with DAgger training data. When combined, our contributions, CIPG and DAgger, provide more stability and lower costs.

5.2. Optimal Behavioral Cloning model structure

Fig. 5 gives the results of the hyperparameter grid search in terms of validation accuracy and model size as presented in Sec. 4.4.1. The worst performers were the feed-forward neural network and random forest, each requiring high memory requirements with only marginal performance increases from more complex models. XGBoost and the LSTM Reverse-time Recurrent Neural Network (RT-RNN) performed similarly, while the GRU RT-RNN performed the best. Therefore, for our final Behavioral Cloning agent we chose the GRU RT-RNN configuration with the highest validation prediction accuracy encircled in Fig. 5.

The optimal model configuration for the selected RT-RNN encircled in Fig. 5 contains one GRU layer with 26 nodes and 7 channels corresponding to each of the parameter groupings that are indexed over the MPC control horizon (Π_2 through Π_7). The previous control values (Π_5) are input to the model through a 1-node layer with ReLU activation function [50]. The outputs of these layers are concatenated with the building model parameters (Π_1) and connected to a 25 node fully connected layer with ReLU activation function. It is then connected to the output layer with sigmoid activation to give the binary control value prediction. Other training hyperparameters are summarized in Tab. 1.

5.3. Final Control Simulation Results

Using the selected optimal RT-RNN model configuration, we analyzed the control performance compared to a baseline thermostat control and the target true MPC control for two scenarios: (1)

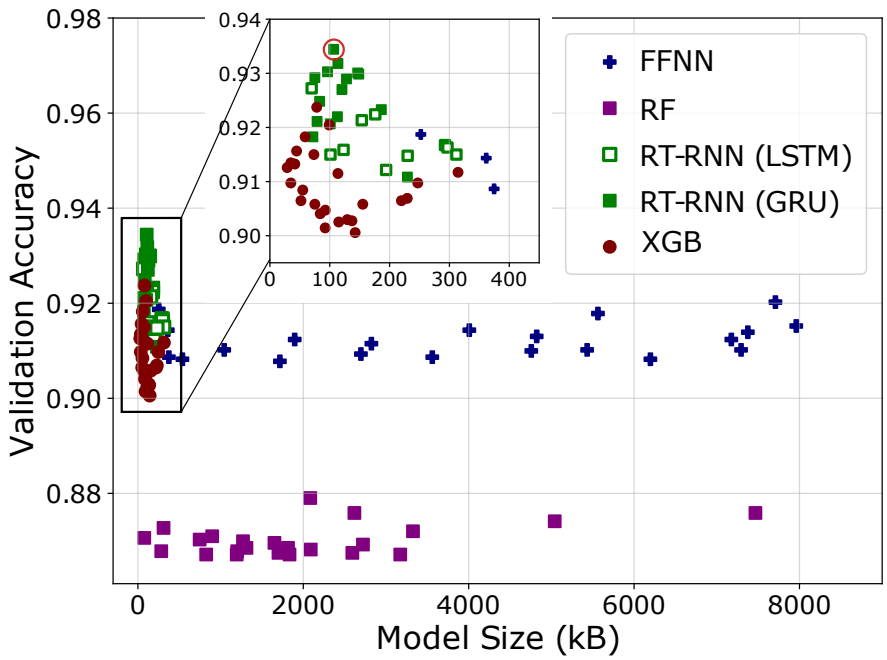


Figure 5: Validation accuracy versus model size for each of the four tested machine learning algorithms. The reverse-time RNN (RT-RNN) models largely outperform the other models on both metrics: It can maintain the highest prediction accuracy with a very small memory requirement. The circled marker denotes the chosen model.

Table 1: Results and training parameters for the optimal RT-RNN configuration

Model Type	GRU
Batch Size	512
Optimizer	Adam [51]
Training Epochs	24
Model Size	106 kB
Validation Accuracy	94.5%

heating in New York City and (2) cooling in Denver, CO. For heating performance, we use the more common time-of-use electricity rate structure with tiers for off-, mid-, and on-peak. For cooling, we use a more variable real-time price based on the day-ahead market that changes every hour [52]. Tab. 2 gives the average processing time, memory requirements, and the average per building electricity cost and comfort violation on the test conditions. The simulations were computed on a Raspberry Pi Zero, which contains a 1GHz single core processor with 512 MB of RAM. Behavioral cloning only requires .1% of the memory of MPC and can operate around 93,000x faster, all while maintaining a similarly low electricity cost and only a modest increase in comfort violations. Moreover, on average the Raspberry Pi, which contains more computing hardware than a typical PLC, was unable to even solve the MPC within the required time step (300 seconds).

Fig. 6 depicts each building's percent improvement in electricity cost and thermal comfort for Behavioral Cloning and MPC compared to the baseline rule-based approach. On average, MPC

Table 2: Control Performance for 10 buildings over a five-day span in both heating and cooling seasons computed on a Raspberry Pi Zero.

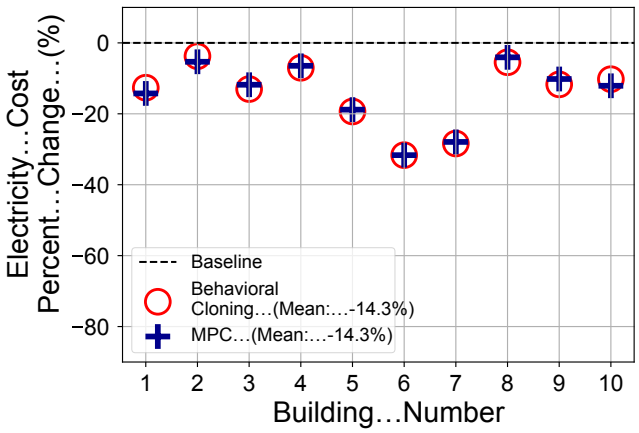
Policy	Scenario	Objective Value	Electricity Cost	Comfort Violation	Memory Requirement	Computational Time Per Step
Rule-Based (baseline)	Heating Cooling	270.79 179.27	\$49.45 \$108.06	221.34 71.21	~ 0	5e-4 s
Behavioral Cloning	Heating Cooling	161.12 124.08	\$42.31 \$82.78	118.81 41.30	176 kB	3.3e-3 s
MPC (target)	Heating Cooling	144.04 119.24	\$42.46 \$77.96	101.58 41.27	150,000 kB	309 s

and Behavioral Cloning perform similarly, with broad improvements to both electricity cost and thermal comfort compared to the baseline. These improvements can vary significantly from building to building based on the setpoint schedules and how well the building is insulated.

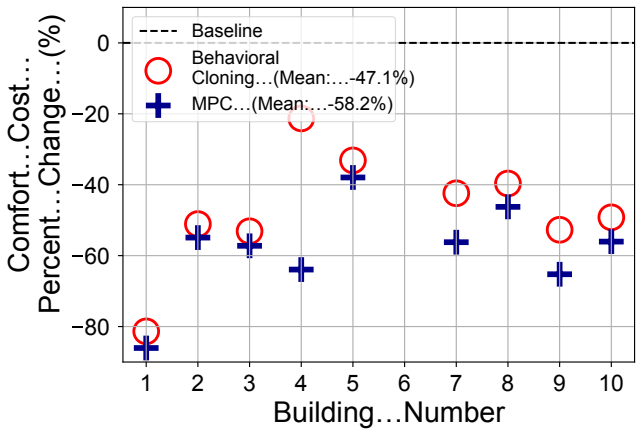
Fig. 7 presents the temperature trajectories for a representative sample of the buildings during winter for each of the control policies: baseline rule-based control, Behavioral Cloning, and MPC. This sample shows the various operating conditions that occur in the overall simulation: small and large setpoint changes and small and large amounts of time when the resident is away. Similar to MPC, Behavioral Cloning maintains the temperature within the lower range of the acceptable thermal comfort band, while still able to effectively preheat the building in preparation for large setpoint changes. These control plots emphasize that though the Behavioral Cloning does not contain any explicit thermodynamic equations or solve any optimization problem, it is able to generalize to new operating conditions and changing user preferences like that of MPC. Each of these setpoint schedules and building-heat pump thermodynamics were not originally included in the training dataset.

6. Conclusion

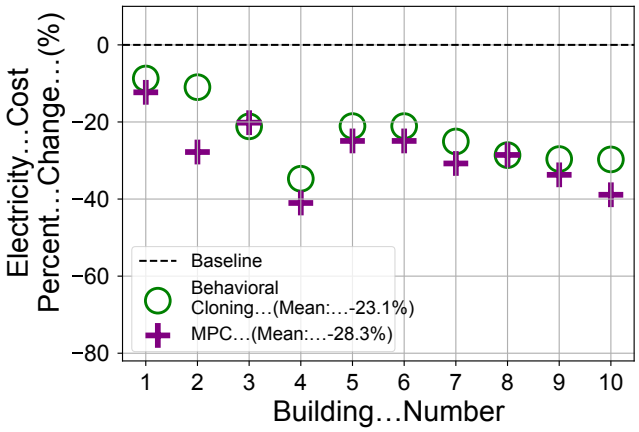
In this paper, we have presented a highly scalable and easy-to-install method for implementing Behavioral Cloning of model predictive control (MPC) on low-cost hardware in many different residential buildings. Our method significantly reduces the installation effort and cost compared to previous approximate MPC studies by requiring the Behavioral Cloning agent to be trained only once for many buildings and operating conditions, rather than needing to be retrained for each specific building it will be implemented on. In addition, our method can adapt to new setpoint schedules and different time-of-use electricity prices, which consistently occur in online operation.



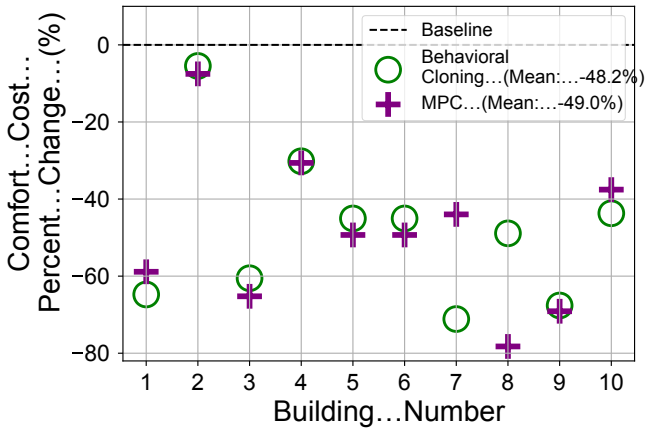
(a) Heating Season



(b) Heating Season



(c) Cooling Season



(d) Cooling Season

Figure 6: Despite requiring orders of magnitude less computational time, objective values improvements over the baseline rule-based approach for each building using Behavioral Cloning show similar performance improvements to that of MPC. Differences in the benefits between buildings are due to varying setpoint schedules and the level of building insulation. Note that Building 6 had near zero comfort costs during the heating season for all three controllers, and thus the percent change is not included.

Simulation results across a range of building parameters, setpoint schedules, and electricity price schedules show that our method provides very similar average efficiency and thermal comfort improvements to that of MPC. Finally, our method only requires .1% of the memory requirements of conventional MPC and can provide the optimal control around 93,000x faster, drastically reducing the computational hardware cost for implementation.

Encouraging building owners to retrofit fossil-fuel systems in favor of heat pumps and to adopt smart building climate control has been, and will likely continue to be, a challenging problem. High capital costs combined with building owners' lack of sufficient knowledge act as barriers to more widespread adoption of clean and efficient heating and cooling. Our method for Behavioral Cloning of MPC can potentially mitigate these barriers by providing a low-cost plug-and-play solution for efficient and flexible heating and cooling control.

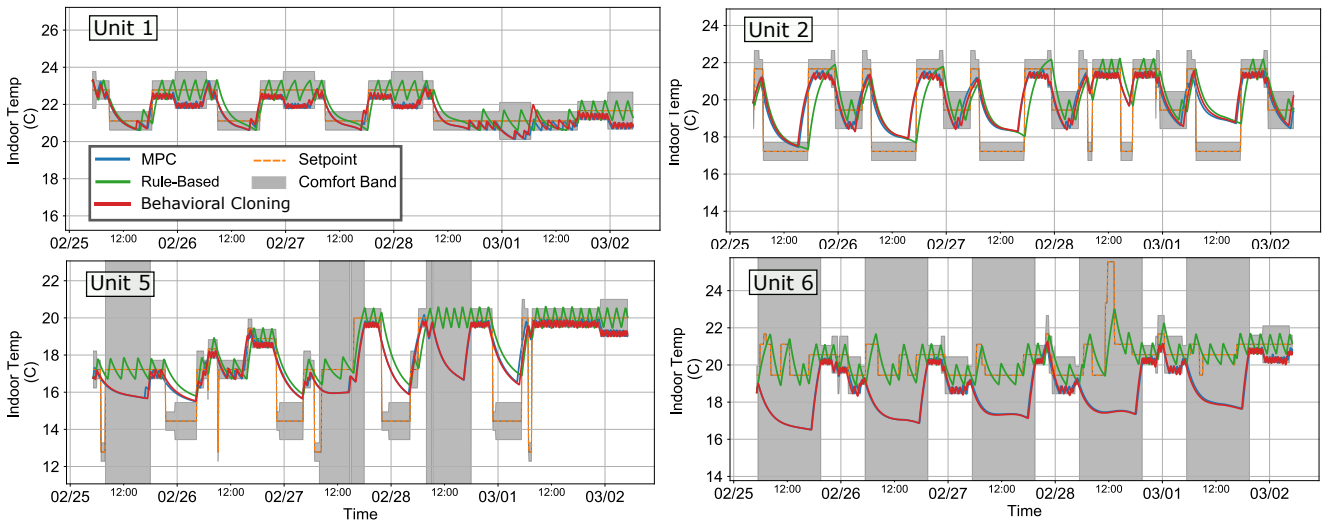


Figure 7: Test control plots for a representative sample of the buildings under each control method for the heating season. Both MPC and Behavioral Cloning can more effectively take advantage of varying setpoint schedules and comfort bands by reducing consumption during times the resident is away, and optimally preheating to avoid comfort violations.

Acknowledgments

The authors acknowledge the support from the National Science Foundation (NSF) under grant 1711546, the NSF Graduate Research Fellowships Program (to ZEL).

References

[1] Z. E. Lee, Q. Sun, Z. Ma, J. Wang, J. S. MacDonald, K. Max Zhang, Providing Grid Services With Heat Pumps: A Review, ASME Journal of Engineering for Sustainable Buildings and Cities 1 (2020). 011007.

[2] M. R. Shaner, S. J. Davis, N. S. Lewis, K. Caldeira, Geophysical constraints on the reliability of solar and wind power in the United States, Energy and Environmental Science 11 (2018) 914–925.

[3] A. Afram, F. Janabi-Sharifi, Theory and applications of HVAC control systems – A review of model predictive control (MPC), Building and Environment 72 (2014) 343–355.

[4] G. Serale, M. Fiorentini, A. Capozzoli, D. Bernardini, A. Bemporad, Model predictive control (MPC) for enhancing building and HVAC system energy efficiency: Problem formulation, applications and opportunities, Energies 11 (2018) 631.

[5] Z. E. Lee, K. M. Zhang, Scalable identification and control of residential heat pumps: A minimal hardware approach, Applied Energy 286 (2021) 116544.

[6] C. Finck, R. Li, W. Zeiler, Economic model predictive control for demand flexibility of a residential building, Energy 176 (2019) 365–379.

[7] J. Cígler, D. Gyalistras, J. Široky, V. Tiet, L. Ferkl, Beyond theory: the challenge of implementing model predictive control in buildings, in: Proceedings of 11th Rehva world congress, Clima, volume 250, 2013.

[8] US Energy Information Administration, Residential Energy Consumption Survey, https://www.eia.gov/consumption/residential/data/2015/index.php/, 2018.

[9] "Smart Thermostats Gain Traction in Europe and North America., Berg Insight, 2017.

[10] Energy Management: Navigating the headwinds, Deloitte Resources, 2016.

[11] G. Hernandez, O. Arias, D. Buentello, Y. Jin, Smart nest thermostat: A smart spy in your home, bit.ly/2XXxrrm, 2014.

[12] J. Drgoňa, M. Kvasnica, M. Klaučo, M. Fikar, Explicit stochastic MPC approach to building temperature control, in: 52nd IEEE Conference on Decision and Control, 2013, pp. 6440–6445.

[13] A. Parisio, L. Fabietti, M. Molinari, D. Varagnolo, K. H. Johansson, Control of hvac systems via scenario-based explicit mpc, in: 53rd IEEE Conference on Decision and Control, 2014, pp. 5201–5207.

[14] Z. E. Lee, K. Gupta, K. J. Kircher, K. M. Zhang, Mixed-integer model predictive control of variable-speed heat pumps, Energy and Buildings 198 (2019) 75–83.

[15] J. Drgoňa, D. Picard, M. Kvasnica, L. Helsen, Approximate model predictive building control via machine learning, Applied Energy 218 (2018) 199–216.

[16] D. B. Crawley, C. O. Pedersen, L. K. Lawrie, F. C. Winkelmann, EnergyPlus: Energy Simulation Program, ASHRAE Journal 42 (2000) 49–56.

[17] S. Yang, M. P. Wan, W. Chen, B. F. Ng, S. Dubey, Experiment study of machine-learning-based approximate model predictive control for energy-efficient building control, Applied Energy 288 (2021) 116648.

[18] T. Wei, Y. Wang, Q. Zhu, Deep reinforcement learning for building HVAC control, in: Proceedings of the 54th Annual Design Automation Conference 2017, DAC '17, Association for Computing Machinery, New York, NY, USA, 2017. URL: https://doi.org/10.1145/3061639.3062224.

[19] Z. Zhang, A. Chong, Y. Pan, C. Zhang, K. P. Lam, Whole building energy model for hvac optimal control: A practical framework based on deep reinforcement learning, Energy and Buildings 199 (2019) 472 – 490.

[20] Z. Wang, T. Hong, Reinforcement learning for building controls: The opportunities and challenges, Applied Energy 269 (2020) 115036.

[21] C. Sammut, S. Hurst, D. Kedzier, D. Michie, Learning to fly, in: Machine Learning Proceedings 1992, Elsevier, 1992, pp. 385–393.

[22] J. Zhang, K. Cho, Query-efficient imitation learning for end-to-end autonomous driving, 2016. arXiv:1605.06450.

[23] I. Bratko, T. Urbančič, C. Sammut, Behavioural cloning: Phenomena, results and problems, IFAC Proceedings Volumes 28 (1995) 143–149. 5th IFAC Symposium on Automated Systems Based on Human Skill (Joint Design of Technology and Organisation), Berlin, Germany, 26-28 September.

[24] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al., End to end learning for self-driving cars, arXiv preprint arXiv:1604.07316 (2016).

[25] J. Carius, F. Farshidian, M. Hutter, MPC-Net: A First Principles Guided Policy Search, IEEE Robotics and Automation Letters 5 (2020) 2897–2904.

[26] F. Torabi, G. Warnell, P. Stone, Behavioral cloning from observation, arXiv preprint arXiv:1805.01954 (2018).

[27] M. Hertneck, J. Kohler, S. Trimpe, F. Allgower, Learning an Approximate Model Predictive Controller with Guarantees, IEEE Control Systems Letters 2 (2018) 543–548.

[28] S. Ross, G. Gordon, D. Bagnell, A reduction of imitation learning and structured prediction to no-regret online learning, in: Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 627–635.

[29] A. Vlachos, An investigation of imitation learning algorithms for structured prediction, in: European Workshop on Reinforcement Learning, 2013, pp. 143–154.

[30] X. Li, J. Wen, Review of building energy modeling for control and operation, Renewable and Sustainable Energy Reviews 37 (2014) 517–537.

[31] CAISO, CAISO Flexible Ramping Product, CAISO, 2016.

[32] N. Good, E. Karangelos, A. Navarro-Espinosa, P. Mancarella, Optimization under uncertainty of thermal storage-based flexible demand response with quantification of residential users' discomfort, IEEE Transactions on Smart Grid 6 (2015) 2333–2342.

[33] Y. Wang, K. Velswamy, B. Huang, A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems, Processes 5 (2017).

[34] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.

[35] A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schlüter, H. Ney, A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 2462–2466.

[36] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555 (2014).

[37] J. Schmidhuber, S. Hochreiter, Long short-term memory, Neural Compute 9 (1997) 1735–1780.

[38] Y. Yu, X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: Lstm cells and network architectures, Neural computation 31 (2019) 1235–1270.

[39] J. Date, J. A. Candanedo, A. k. Athienitis, Control-oriented modelling of thermal zones in a house: a multi-level approach, in: International High Performance Buildings Conference, 2016. URL: `http://docs.lib.purdue.edu/ihpbc/229`.

[40] B. Eisenhower, Z. O'Neill, S. Narayanan, V. A. Fonoberov, I. Mezić, A methodology for meta-model based optimization in building energy models, Energy and Buildings 47 (2012) 292–301.

[41] D. H. Blum, K. Arendt, L. Rivalin, M. A. Piette, M. Wetter, C. T. Veje, Practical factors of envelope model setup and their effects on the performance of model predictive control for building heating, ventilating, and air conditioning systems, Applied Energy 236 (2019) 410–425.

[42] Technical guide LX Series Split System Heat Pumps, 2019.

[43] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[44] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[45] Ecobee Inc., Donate your data, 2019. `https://www.ecobee.com/donateyourdata/`.

[46] New York State Electric and Gas Corporation, Electricity service rate, 2018. Service class 12, Rate No. 115-12-00.

[47] Consolidated Edison, Inc., Residential Time-of-Use Rate, 2020. Https://bit.ly/3cJIRF5.

[48] Xcel Energy, Time of Use Pricing, 2020. Https://bit.ly/3cUraTr.

[49] M. Sengupta, Y. Xie, A. Lopez, A. Habte, G. Maclaurin, J. Shelby, The National Solar Radiation Data Base (NSRDB), Renewable and Sustainable Energy Reviews 89 (2018) 51 – 60.

[50] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 315–323.

[51] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[52] New York ISO, Day-Ahead Market LBMP, 2020. Https://www.nyiso.com/custom-reports.