One for Many: Transfer Learning for Building HVAC Control

Shichao Xu
Northwestern University
Evanston, USA
shichaoxu2023@u.northwestern.edu

Yixuan Wang
Northwestern University
Evanston, USA
yixuanwang2024@u.northwestern.edu

Yanzhi Wang
Northeastern University
Boston, USA
yanz.wang@northeastern.edu

Zheng O'Neill
Texas A&M University
College Station, USA
zoneill@tamu.edu

Qi Zhu
Northwestern University
Evanston, USA
qzhu@northwestern.edu

ABSTRACT

The design of building heating, ventilation, and air conditioning (HVAC) system is critically important, as it accounts for around half of building energy consumption and directly affects occupant comfort, productivity, and health. Traditional HVAC control methods are typically based on creating explicit physical models for building thermal dynamics, which often require significant effort to develop and are difficult to achieve sufficient accuracy and efficiency for runtime building control and scalability for field implementations. Recently, deep reinforcement learning (DRL) has emerged as a promising data-driven method that provides good control performance without analyzing physical models at runtime. However, a major challenge to DRL (and many other data-driven learning methods) is the long training time it takes to reach the desired performance. In this work, we present a novel transfer learning based approach to overcome this challenge. Our approach can effectively transfer a DRL-based HVAC controller trained for the source building to a controller for the target building with minimal effort and improved performance, by decomposing the design of neural network controller into a transferable front-end network that captures building-agnostic behavior and a back-end network that can be efficiently trained for each specific building. We conducted experiments on a variety of transfer scenarios between buildings with different sizes, numbers of thermal zones, materials and layouts, air conditioner types, and ambient weather conditions. The experimental results demonstrated the effectiveness of our approach in significantly reducing the training time, energy cost, and temperature violations.

CCS CONCEPTS

• **Computing methodologies → Reinforcement learning**; • **Computer systems organization → Embedded and cyber - physical systems**.

KEYWORDS

Smart Buildings, HVAC control, Data-driven, Deep reinforcement learning, Transfer learning

1 INTRODUCTION

The building stock accounts for around 40% of the annual energy consumption in the United States, and nearly half of the building energy is consumed by the heating, ventilation, and air conditioning (HVAC) system [26]. On the other hand, average Americans spend approximately 87% of their time indoors [15], where the operation of HVAC system has a significant impact on their comfort, productivity, and health. Thus, it is critically important to design HVAC control systems that are both energy efficient and able to maintain the desired temperature and indoor air quality for occupants.

In the literature, there is an extensive body of work addressing the control design of building HVAC systems [20, 27, 30, 33]. Most of them use **model-based** approaches that create simplified physical models to capture building thermal dynamics for efficient HVAC control. For instance, resistor-capacitor (RC) networks are used for modeling building thermal dynamics in [20–22], and linear-quadratic regulator (LQR) or model predictive control (MPC) based approaches are developed accordingly for efficient runtime control. However, creating a simplified yet sufficiently-accurate physical model for runtime HVAC control is often difficult, as building room air temperature is complexly affected by a number of factors, including building layout, structure, construction and materials, surrounding environment (e.g., ambient temperature, humidity, and solar radiation), internal heat generation from occupants, lighting, and appliances, etc. Moreover, it takes significant effort and time to develop explicit physical models, find the right parameters, and update the models over the building lifecycle [28].

The drawbacks of model-based approaches have motivated the development of **data-driven** HVAC control methods that do not rely on analyzing physical models at runtime but rather directly making the decisions based on input data. A number of data-driven methods such as reinforcement learning (RL) have been proposed in the literature, including more traditional methods that leverage the classical Q-learning techniques and perform optimization based on a tabular Q value function [2, 16, 25], earlier works that utilize neural networks [4, 7], and more recent deep reinforcement learning (DRL) methods [8, 9, 17, 24, 29, 36, 37]. In particular, the DRL-based methods leverage deep neural networks for estimating the Q values associated with state-action pairs and are able to handle larger state space than traditional RL methods [28]. They have emerged as a promising solution that offers good HVAC control performance without analyzing physical models at runtime.

However, there are major challenges in deploying DRL-based methods in practice. Given the complexity of modern buildings, it could take a significant amount of training for DRL models to reach the desired performance. For instance, around 50 to 100 months of data are needed for training the models in [28, 29] and 4000+ months of data are used for more complex models [9, 34] – even if this could be drastically reduced to a few months or weeks, directly

deploying DRL models on operational buildings and taking so long before getting the desired performance is impractical. The works in [28, 29] thus propose to first use detailed and accurate physical models (e.g., EnergyPlus [5]) for offline simulation-based training before the deployment. While such an approach can speed up the training process, it still requires the development and update of detailed physical models, which as stated above needs significant domain expertise, effort, and time.

To address the challenges in DRL training for HVAC control, we propose a **transfer learning** based approach in this paper, to utilize existing models (that had been trained for old buildings) in the development of DRL methods for new buildings. This is not a straightforward process, however. Different buildings may have different sizes, numbers of thermal zones, materials and layouts, HVAC equipment, and operate under different ambient weather conditions. As shown later in the experiments, directly transferring models between such different buildings is not effective. In the literature, there are a few works that have explored transfer learning for buildings. In [3], a building temperature and humidity prediction model is learned from supervised learning, and transferred to new buildings with further tuning and utilized in an MPC algorithm. The work in [18] investigates the transfer of Q-learning for building HVAC control under different weather conditions and with different room sizes, but it is limited to single-room buildings. The usage of Q-table in conventional Q-learning also leads to limited memory for state-action pairs and makes it unsuitable for complex buildings.

Our work addresses the limitations in the literature, and develops for the first time a Deep Q-Network (DQN) based transfer learning approach for multiple-zone buildings. Our approach avoids the development of physical models, significantly reduces the DRL training time via transfer learning, and is able to reduce energy cost while maintaining room temperatures within desired bounds. More specifically, our work makes the following contributions:

- We propose a novel transfer learning approach that decomposes the design of neural network based HVAC controller into two (sub-)networks. The front-end network captures building-agnostic behavior and can be directly transferred, while the back-end network can be efficiently trained for each specific building in an offline supervised manner by leveraging a small amount of data from existing controllers (e.g., simple on-off controller).

- Our approach requires little to no further tuning of the transferred DRL model after it is deployed in the new building, thanks to the two-subnetwork design and the offline supervised training of the back-end network. This avoids the initial *cold start* period where the HVAC control may be unstable and unpredictable.

- We have performed a number of experiments for evaluating the effectiveness of our approach under various scenarios. The results demonstrate that our approach can effectively transfer between buildings with different sizes, numbers of thermal zones, materials and layouts, and HVAC equipment, as well as under different weather conditions in certain cases. Our approach could enable fast deployment of DRL-based HVAC control with little training time after transfer, and reduce building energy cost with minimal violation of temperature constraints.

The rest of the paper is structured as follows. Section 2 provides a more detailed review of related work. Section 3 presents our approach, including the design of two networks and the corresponding training methods. Section 4 shows the experiments for different transfer scenarios and other related ablation studies. Section 5 concludes the paper.

2 RELATED WORK

Model-based and Data-driven HVAC Control. There is a rich literature in HVAC control design, where the approaches can generally fall into two main categories, i.e., model-based and data-driven.

Traditional model-based HVAC control approaches typically build explicit physical models for the controlled buildings and their surrounding environment, and then design control algorithms accordingly [20, 27]. For instance, the work in [19] presents a nonlinear model for the overall cooling system, which includes chillers, cooling towers and thermal storage tanks, and then develops an MPC-based approach for reducing building energy consumption. The work in [20] models the building thermal dynamics as RC networks, calibrates the model based on historical data, and then presents a tracking LQR approach for HVAC control. Similar simplified models have been utilized in other works [21, 22, 30] for HVAC control and for co-scheduling HVAC operation with other energy demands and power supplies. While being efficient, these simplified models often do not provide sufficient accuracy for effective runtime control, given the complex relation between building room air temperature and various factors of the building itself (e.g., layout, structure, construction and materials), its surrounding environment (e.g., ambient temperature, humidity, solar radiation), and internal operation (e.g., heat generation from occupants, lighting and appliances). More accurate physical models can be built and simulated with tools such as EnergyPlus [5], but those models are typically too complex to be used for runtime control.

Data-driven approaches have thus emerged in recent years due to their advantages of not requiring explicit physical models at runtime . These approaches often leverage various machine learning techniques, in particular reinforcement learning. For instance, in [29, 37], DRL is applied to building HVAC control and an Energy-Plus model is leveraged for simulation-based offline training of DRL. In [8, 36], DRL approaches leveraging the actor-critic methods are applied. The works in [9, 24] use data-driven methods to approximate/learn the energy consumption and occupants' satisfaction under different thermal conditions, and then apply DRL to learn an end-to-end HVAC control policy. These DRL-based methods are shown to be effective at reducing energy cost and maintaining desired temperature, and are sufficiently efficient at runtime. However, they often take a long training time to reach the desired performance, needing dozens and hundreds of months of data for training [28, 29] or even longer [9, 34]. Directly deploying them in real buildings for such long training process is obviously not practical. Leveraging tools such as EnergyPlus for offline simulation-based training can mitigate this issue, but again incurs the need for the expensive and sometimes error-prone process of developing accurate physical models (needed for simulation in this case). These challenges have motivated this work to develop a transfer learning approach for efficient and effective DRL control of HVAC systems.

Transfer Learning for HVAC control. There are a few works that have explored transfer learning in buildings HVAC control. In [18], transfer learning of a Q-learning agent is studied, however only a single room (thermal zone) is considered. The usage of a tabular table for each state-action pair in the traditional Q-learning in fact limits the approach's capability to handle high-dimensional data. In [3], a neural network model for predicting temperature and humidity is learned in a supervised manner and transferred to new buildings for MPC-based control. The approach also focuses on single-zone buildings and requires further tuning after the deployment of the controller.

Different from these earlier works in transfer learning for HVAC control, our approach addresses multi-zone buildings and considers transfer between buildings with different sizes, number of thermal zones, layouts and materials, HVAC equipment, and ambient weather conditions. It also requires little to no further tuning after the transfer. This is achieved with a novel DRL controller design with two sub-networks and the corresponding training methods.

Transfer Learning in DRL. Since our approach considers transfer learning for DRL, it is worth to note some of the work in DRL-based transfer learning for other domains [1, 6, 11, 35]. For instance, in [11], the distribution of optimal trajectories across similar robots is matched for transfer learning in robotics. In [1], an environment randomization approach is proposed, where DRL agents trained in simulation with a large number of generated environments can be successfully transferred to their real-world applications. To the best of our knowledge, our work is the first to propose DRL-based transfer learning for multi-zone building HVAC control. It addresses the unique challenges in building domain, e.g., designing a novel two-subnetwork controller to avoid the complexity and cost of creating accurate physical models for simulation.

3 OUR APPROACH

We present our transfer learning approach in this section, including the design of the two-subnetwork controller and the training process. Section 3.1 introduces the system model. Section 3.2 provides an overview of our methodology. Section 3.3 presents the design of the building-agnostic front-end (sub-)network, and Section 3.4 explains the design of the building-specific back-end (sub-)network.

3.1 System Model

The goal of our work is to build a transferable HVAC control system that can maintain comfortable room air temperature within desired bounds while reducing the energy cost. We adopt a building model that is similar to the one used in [29], an n-zone building model with a variable air volume (VAV) HVAC system. The system provides conditioned air at a flow rate chosen from m discrete levels. Thus, the entire action space for the n-zone controller can be described as $\mathbf{A} = \{\mathbf{a_1}, \mathbf{a_2}, \cdots, \mathbf{a_n}\}$, where $\mathbf{a_i}(1 \leq i \leq n)$ is chosen from m VAV levels $\{f_1, f_2, \cdots, f_m\}$. Note that the size of the action space (m^n) increases exponentially with respect to the number of thermal zones n, which presents significant challenge to DRL control for larger buildings. We address this challenge in the design of our two-subnetwork DRL controller by avoiding setting the size of the neural network action output layer to m^n. This will be explained further later.

The DRL action is determined by the current system state. In our model, the system state includes the current physical time t, inside state S_{in}, and outside environment state S_{out}. The inside state S_{in} includes the temperature of each thermal zone, denoted as $\{T_1, T_2, \cdots, T_n\}$. The outside environment state S_{out} includes the ambient temperature and the solar irradiance (radiation intensity). Similar to [29], to improve DRL performance, S_{out} not only includes the current values of the ambient temperature T_{out}^i and the solar irradiance Sun_{out}^i, but also their weather forecast values for the next three days. Thus, the outside environment state is denoted as $S_{out} = \{T_{out}^0, T_{out}^1, T_{out}^2, T_{out}^3, Sun_{out}^0, Sun_{out}^1, Sun_{out}^2, Sun_{out}^3\}$. Our current model does not consider internal heat generation from occupants, a limitation that we plan to address in future work.

3.2 Methodology Overview

We started our work by considering whether it is possible to directly transfer a well-trained DQN model for a single-zone source building to every zone of a target multiple-zone building. However, based on our experiments (shown later in Table 2 of Section 4), such straightforward approach is not effective at all, leading to significant temperature violations. This is perhaps not surprising. In DQN-based reinforcement learning, a neural network Q maps the input $I = \{I_1, I_2, \cdots, I_n\}$, where I_i is the state for each zone i, to the control action output $\mathbf{A}$. The network Q is optimized based on a reward function that considers energy cost and temperature violation. Through training, Q learns a control strategy that incorporates the consideration of building thermal dynamics, including the building-specific characteristics. Directly applying Q to a new target building, which may have totally different characteristics and dynamics, will not be effective in general.

Thus, our approach designs a novel architecture that includes two sub-networks, with an intermediate state ΔT that indicates a predictive value of the controller's willingness to change the indoor temperature. The **front-end network** Q maps the inputs I to the intermediate state ΔT. It is trained to capture the building-agnostic part of the control strategy, and is directly transferable. The **back-end network** then maps ΔT, together with I, to the control action output $\mathbf{A}$. It is trained to capture the building-specific part of the control, and can be viewed as an inverse building network F^{-1}. An overview of our approach is illustrated in Figure 1.

3.3 Front-end Building-agnostic Network Design and Training

We introduce the design of our front-end network Q and its training in this section. Q is composed of n (sub-)networks itself, where where n is the number of building thermal zones. Each zone in the building model has its corresponding sub-network, and all sub-networks share their weights. In each sub-network for thermal zone i, the input layer accepts state I_i. It is followed by L sequentially-connected fully-connected layers (the exact number of neurons is presented later in Table 1 of Section 4). Rather than directly giving the control action likelihood vector, the network's output layer reflects a planned temperature change value ΔT_i for each zone.

More specifically, the output of the last layer is designed as a vector $O_{\Delta T_i}$ of length $h + 2$ in one-hot representation – the planned temperature changing range is equally divided into h intervals
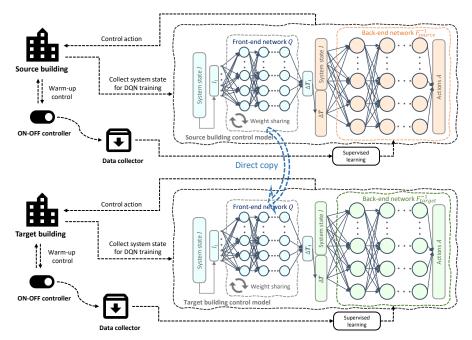
Figure 1: Overview of our DRL-based transfer learning approach for HVAC control. We design a novel DQN architecture that includes two sub-networks: A front-end network Q captures the building-agnostic part of the control as much as possible, while a back-end network (inverse building network) F^{-1} captures the building-specific behavior. At each control step, the front-end network Q maps the current system state I to an intermediate state ΔT. Then, the back-end network F^{-1} maps ΔT, together with I, to the control action outputs A. During transfer learning from a source building to a target building, the front-end network Q is directly transferable. The back-end network F^{-1} can be trained in a supervised manner, with data collected from an existing controller (e.g., a simple ON-OFF controller). Experiments have shown that around two weeks of data is sufficient for such supervised training of F^{-1}. If it is a brand new building without any existing controller, we can deploy a simple ON-OFF controller for two weeks in a "warm-up" process. During this process, the ON-OFF controller can maintain the temperature within the desired bounds (albeit with higher cost), and collect data that captures the building-specific behavior for training F^{-1}.

within a predefined temperature range of $[-b, b]$ and two intervals outside of that range are also considered. The relationship of the planned temperature change value ΔT_i of zone i and the output vector $O_{\Delta T_i}$ is as follows:

$$O_{\Delta T_i} = \begin{cases} < 1, 0, \cdots, 0 >, & \Delta T_i \leq -b, \\ < 0, \cdots, 0, 1, 0, \cdots, 0 >, & -b < \Delta T_i < b, \\ \quad the\ position\ of\ 1\ is\ at\ (\lfloor \Delta T_i/(2b/h) \rfloor) \\ < 0, \cdots, 0, 1 >, & \Delta T_i \geq b. \end{cases}$$
(1)

Then, for the entire front-end network Q, the combined input is $I = \{I_1, I_2, \cdots, I_n\}$, and the combined output is $O_{\Delta T} = \{O_{\Delta T_1}, O_{\Delta T_2}, \cdots, O_{\Delta T_n}\}$.

It is worth noting that if we had designed the front-end network in standard deep Q-learning model [23], it would take I as the network's input, pass it through several fully-connected layers, and output the selection among an action space that has a size of $(h + 2)^n$ (as there are n zones, and each has $h + 2$ possible actions). It also needs an equal number of neurons for the last layer, which is not affordable when the number of zones gets large. Instead in our design, the last layer of the front-end network Q has its size reduced to $(h + 2) * n$, which can be further reduced to $(h + 2)$ with the following weight-sharing technique.

We decide to let the n sub-networks of Q share their weights during training. One benefit of this design is that it enables transferring the front-end network for a n-zone source building to a target m-zone building, where m could be different from n. It also reduces the training load by lowering the number of parameters. Such design performs well in our experiments.

Our front-end network Q is trained with the standard deep Q-learning techniques [23]. Note that while the output action for Q is the planned temperature change vector $O_{\Delta T}$, the training process uses a dynamic reward R_t that depends on the eventual action (i.e., output of network F^{-1}), which will be introduced later in Section 3.4. Specifically, the training of the front-end network Q follows Algorithm 1 (the hyper-parameters used are listed later in Table 1 of Section 4). First, we initialize Q by following the weights initialization method described in [12] and copy its weights to the target network Q' (target network Q' is a technique in deep Q-learning that is used for improving performance.). The back-end network F^{-1} is initialized following Algorithm 2 (introduced later in Section 3.4). We also empty the replay buffer and set the exploration rate ϵ to 1.

At each control instant t during a training epoch, we obtain the current system state $S_{cur} = (t, S_{in}, S_{out})$ and calculate the current reward R_t. We then collect the learning samples (experience) $(S_{pre}, S_{cur}, \Delta T, \mathbf{A}, R)$ and store them in the replay buffer. In the following learning-related operations, we first sample a data batch

$M = (\mathcal{S}_{prime}, \mathcal{S}_{next}, a, r)$ from the replay buffer, and calculate the actual temperature change value $\Delta \mathcal{T}_a$ from $\mathcal{S}_{prime}$ and $\mathcal{S}_{next}$. Then, we get the planned temperature change value from the back-end network F^{-1}, i.e., $a_p = F^{-1}(\Delta \mathcal{T}_a, \mathcal{S}_{prime})$. In this way, the cross entropy loss can be calculated from the true label a and the predicted label a_p. We then use supervised learning to update the back-end network F^{-1} with the Adam optimizer [14] under learning rate lr_2.

We follow the same procedure as described in [23] to calculate the target vector v that is used in deep Q-learning. With target vector v and input state $\mathcal{S}_{prime}$, we can then train Q using the back-propagation method [10] with mean squared error loss and learning rate lr_1. With a period of Δnt, we assign the weights of Q to the target network Q'. The exploration rate is updated as $\epsilon = \max\{\epsilon_{low}, \epsilon - \Delta\epsilon\}$. It is used for ϵ−greedy policy to select each planned temperature change value ΔT_i:

$$\Delta T_i = \begin{cases} argmax\ O_{\Delta T_i} & with\ probability\ 1 - \epsilon, \\ random(0\ to\ h+1) & with\ probability\ \epsilon. \end{cases} \quad (2)$$

$$\Delta T = \{\Delta T_1, \Delta T_2, \cdots, \Delta T_n\}. \quad (3)$$

The control action $\mathbf{A}$ is obtained from the back-end network:

$$\mathbf{A} = F^{-1}(\Delta T, S_{cur}). \quad (4)$$

3.4 Back-end Building-specific Network Design and Training

The objective of the back-end network is to map the planned temperature change vector $O_{\Delta T}$ (or ΔT), together with the system state I, into the control action $\mathbf{A}$. Consider that during operation, a building environment "maps" the control action and system state to the actual temperature change value. So in a way, the back-end network can be viewed as doing the *inverse* of what a building environment does, i.e., it can be viewed as an inverse building network F^{-1}.

The network F^{-1} receives the planned temperature change value ΔT and the system state I at its input layer. It is followed by L' fully-connected layers (exact number for experimentation is specified in Table 1 of Section 4). It outputs a likelihood control action vector $O_{\mathbf{A}} = \{v_1, v_2, \cdots, v_n\}$, which can be divided into n groups. For group i, it has a one-hot vector v_i corresponding to the control action for zone i. The length of v_i is m, as there are m possible control actions for each zone as defined earlier. When $O_{\mathbf{A}}$ is provided, control action $\mathbf{A}$ can be easily calculated by applying argmax operation for each group in $O_{\mathbf{A}}$, i.e., $\mathbf{A} = \{argmax\{v_1\}, argmax\{v_2\}, \cdots, argmax\{v_n\}\}$.

The network F^{-1} is integrated with the reward function R_t:
$$R_t = w_{cost}R\_cost_t + w_{vio}R\_vio_t, \quad (5)$$
where R_cost_t is the reward of energy cost at time step t and w_{cost} is the corresponding scaling factor. R_vio_t is the reward of zone temperature violation at time step t and w_{vio} is its scaling factor. The two rewards are further defined as:

$$R\_cost_t = -cost(F^{-1}(\Delta T_{t-1}), t-1). \quad (6)$$

$$R\_vio_t = -\sum_{i=1}^{n} max(T_t^i - T_{upper}, 0) + max(T_{lower} - T_t^i, 0). \quad (7)$$

Algorithm 1 Training of front-end network Q

1: ep: the number of training epochs
2: Δct: the control period
3: t_{MAX}: the maximum training time of an epoch
4: Δnt: the time interval to update target network
5: Empty replay buffer
6: Initialize Q; set the weights of target network $Q' = Q$; initialize F^{-1} based on Algorithm 2
7: Initialize the current planned temperature change vector ΔT
8: Initialize previous state S_{pre}
9: Initialize exploration rate ϵ
10: **for** $Epoch$ = 1 to ep **do**
11:     **for** t = 0 to t_{MAX}, t += Δct **do**
12:         $S_{cur} \leftarrow (t, S_{in}, S_{out})$
13:         Calculate reward R
14:         Add experience $(S_{pre}, S_{cur}, \Delta T, \mathbf{A}, R)$ to the replay buffer
15:         **for** tr = 0 to L_{MAX} **do**
16:             Sample a batch $M = (\mathcal{S}_{prime}, \mathcal{S}_{next}, a, r)$
17:             Calculate actual temperature change value $\Delta \mathcal{T}_a$
18:             Predicted label $a_p = F^{-1}(\Delta \mathcal{T}_a, \mathcal{S}_{prime})$
19:             Set loss $L = CrossEntropyLoss(a_p, a)$
20:             Update F^{-1} with loss L and learning rate lr_2
21:             Target $v \leftarrow$ target network $Q'(\mathcal{S}_{prime})$
22:             Train network Q with $\mathcal{S}_{prime}$ and v
23:         **end for**
24:         **if** t mod Δnt == 0 **then**
25:             Update target network Q'
26:         **end if**
27:         $O_{\Delta T} = Q(S_{cur})$
28:         Update exploration rate ϵ
29:         Update each ΔT_i follows ϵ−greedy policy
30:         $\Delta T = < \Delta T_1, \Delta T_2, \cdots, \Delta T_n >$
31:         Control action $\mathbf{A} \leftarrow F^{-1}(\Delta T, S_{cur})$
32:         $S_{pre} = S_{cur}$
33:     **end for**
34: **end for**

Here, $cost(,)$ is a function that calculates the energy cost within a control period according to the local electricity price that changes over time. ΔT_{t-1} is the planned temperature change value at time $t - 1$. T_t^i is the zone i temperature at time t. T_{upper} and T_{lower} are the comfortable temperature upper / lower bound, respectively.

As stated before, F^{-1} can be trained in a supervised manner. We could also directly deploy our DRL controller, with transferred front-end network Q and an initially-randomized back-end network F^{-1}; but we have found that leveraging data collected from the existing controller of the target building for offline supervise learning of F^{-1} before deployment can provide significantly better results than starting with a random F^{-1}. This is because that the data from the existing controller provides insights into the building-specific behavior, which after all is what F^{-1} is for. In our experiments, we have found that a simple existing controller such as the ON-OFF controller with two weeks of data can already be very effective for helping training F^{-1}. Note that such supervised training of F^{-1} does not require the front-end network Q, which means F^{-1} could be well-trained and ready for use before Q is trained and transferred. In the case that the target building is brand new and there is no existing controller, we can deploy a simple ON-OFF controller for collecting such data in a warm-up process (Figure 1). While such

Algorithm 2 Training of back-end network F^{-1}
___
1: ep_F: the number of training epochs
2: Δct: the control period
3: t'_{MAX}: the maximum data collection time
4: Initialize previous state S_{pre}
5: Initialize F^{-1}
6: Empty database M and dataset D
7: **for** $t = 0$ to t_{MAX}, t += Δct **do**
8:     $S_{cur} \leftarrow (t, S_{in}, S_{our})$
9:     Control action $\mathbf{A} \leftarrow$ run ON-OFF controller on S_{cur}
10:     $S_{pre} = S_{cur}$
11:     Add sample $(S_{cur}, S_{pre}, \mathbf{A})$ to database M
12: **end for**
13: **for** each sample $\mathbf{u} = (S_{cur}, S_{pre}, \mathbf{a})$ in M **do**
14:     $\Delta \mathcal{T}_a \leftarrow$ calculate temperature difference in $(\mathcal{S}_{cur}, \mathcal{S}_{pre})$
15:     Add sample $\mathbf{v} = (\Delta \mathcal{T}_a, S_{pre}, \mathbf{a})$ to dataset D
16: **end for**
17: **for** each sample $\mathbf{u} = (S_{cur}, S_{pre}, \mathbf{a})$ in M **do**
18:     $\Delta \mathcal{T}_a \leftarrow$ lowest level
19:     $\mathbf{a}' \leftarrow$ maximum air condition level
20:     Add sample $\mathbf{v} = (\Delta \mathcal{T}_a, S_{pre}, \mathbf{a}')$ to dataset D
21: **end for**
22: **for** $Epoch = 1$ to ep_F **do**
23:     **for** each training batch of $(\Delta \mathcal{T}_a, S_{pre}, \mathbf{a})$ in dataset D **do**
24:         network inputs = $(\Delta \mathcal{T}_a, S_{pre})$
25:         corresponding labels = $(\mathbf{a})$
26:         Train network F^{-1}
27:     **end for**
28: **end for**
29: Return F^{-1}
___

Algorithm 3 Running of our proposed approach
___
1: Δct: the control period
2: t_{MAX}: the maximum testing time
3: Initialize the weights of Q with the front-end network transferred from the source building (see Figure 1)
4: Initialize the weights of F^{-1} with weights learned using Algorithm 2
5: **for** $t = 0$ to t_{MAX}, t += Δct **do**
6:     $S_{cur} \leftarrow (t, S_{in}, S_{out})$
7:     $\Delta T \leftarrow argmax\ Q(S_{cur})$
8:     Control action $\mathbf{A} \leftarrow F^{-1}(\Delta T, S_{cur})$
9: **end for**
___

ON-OFF controller typically consumes significantly higher energy, it can effectively maintain the room temperature within desired bounds, which means that the building could already be in use during this period. Once F^{-1} is trained, the DRL controller can replace the ON-OFF controller in operation.

Algorithm 2 shows the detailed process for the training of F^{-1}. Note that the initialization of F^{-1} in this algorithm also follows the weights initialization method described in [12]. We also augment the collected training data to ensure the boundary condition. The augmenting data is created by copying all samples from the collected data and set temperature change value $\Delta \mathcal{T}$ to the lowest level ($< -b$) while setting all control actions to the maximum level.

Once the front-end network Q is trained as in Algorithm 1 and the back-end network F^{-1} is trained as in Algorithm 2, our transferred DRL controller is ready to be deployed and can operate as

Parameter	Value	Parameter	Value
Front-end network layers	[10,128,256, 256,256,400,22]	Back-end network layers	[22*n,128,256, 256,128,m*n]
b	2	h	20
lr_1	0.0003	ep	150
lr_2	0.0001	ep_F	15
L_{MAX}	1	w_{cost}	$\frac{1}{1000}$
ep	150	w_{vio}	$\frac{1}{1600}$
T_{lower}	19	T_{upper}	24
Δnt	240*15 min	Δct	15 min
t'_{MAX}	2 weeks	t_{MAX}	1 month
ϵ_{low}	0.1		

Table 1: Hyper-parameters used in our experiments.

described in Algorithm 3. Note that we could further fine-tune our DRL controller during the operation. This can be done by enabling a fine-tuning procedure that is similar to Algorithm 1. The difference is that instead of initializing the Q-network Q using [12], we copy transferred Q-network weights from the source building to the target building's front-end network Q and its corresponding target network Q'. And we set $\epsilon = 0$, $\epsilon_{low} = 0$, and L_{MAX} to 3 instead of 1. Other operations remain the same as in Algorithm 1.

4 EXPERIMENTAL RESULTS

4.1 Experiment Settings

All experiments are conducted on a server equipped with a 2.10GHz CPU (Intel Xeon(R) Gold 6130), 64GB RAM, and an NVIDIA TITAN RTX GPU card. The learning algorithms are implemented in the PyTorch learning framework. The Adam optimizer [14] is used to optimize both front-end networks and back-end networks. The DRL hyper-parameter settings are shown in Table 1. In addition, to accurately evaluate our approach, we leverage the building simulation tool EnergyPlus [5]. Note that EnergyPlus here is only used for evaluation purpose, in place of real buildings. During the practical application of our approach, EnergyPlus is not needed. This is different from some of the approaches in the literature [28, 29], where EnergyPlus is needed for offline training before deployment and hence accurate and expensive physical models have to be developed.

In our experiments, simulation models in EnergyPlus interact with the learning algorithms written in Python through the Building Controls Virtual Test Bed (BCVTB) [31]. We simulate the building models with the weather data obtained from the Typical Meteorological Year 3 database [32], and choose the summer weather data in August (each training epoch contains one-month data). Apart from the weather transferring experiments, all other experiments are based on the weather data collected in Riverside, California, where the ambient weather changes more drastically and thus presents more challenges to the HVAC controller. Different building types are used in our experiments, including one-zone building 1 (simplified as *1-zone 1*), four-zone building 1 (*4-zone 1*), four-zone building 2 (*4-zone 2*), four-zone building 3 (*4-zone 3*), five-zone building 1 (*5-zone 1*), seven-zone building 1 (*7-zone 1*). These models are visualized in Figure 2. In addition, the conditioned air temperature sent from the VAV HVAC system is set to 10 ℃.

The symbols used in the result tables are explained as follows. θ_i denotes the temperature violation rate in the thermal zone i. $A\theta$ and $M\theta$ represent the average temperature violation rate across

all zones and the maximum temperature violation rate across all zones, respectively. μ_i denotes the maximum temperature violation value for zone i, measured in ℃. Aμ and Mμ are the average and maximum temperature violation value across all zones, respectively. EP represents the number of training epochs. The symbol ☑ denotes whether all the temperature violation rates across all zones are less than 5%. If it is true, it is marked as ✓; otherwise, it is × (which is typically not acceptable for HVAC control).

Before reporting the main part of our results, we want to show that simply transferring a well-trained DQN model for a single-zone source building to every zone of a target multi-zone building may not yield good results, as discussed in Section 3.2. Here as shown in Table 2, a DQN model trained for one-zone building 1 works well for itself, but when being transferred directly to every zone of four-zone building 2, there are significant temperature violations. This shows that a more sophisticated approach such as ours is needed. The following sections will show the results of our approach and its comparison with other methods.

4.2 Transfer from n-zone to n-zone with different materials and layouts

In this section, we conduct experiments on building HVAC controller transfer with four-zone buildings that have different materials and layouts. As shown in Figure 2, four-zone building 1 and four-zone building 2 have different structures, and also different wall materials in each zone with different heat capacities. Table 3 first shows the direct training results on four-zone building 1, and the main transferring results are presented in Table 4.

The direct training outcome by baselines and our approach are shown in Table 3. The results include ON-OFF control, Deep Q-network (DQN) control as described in [29] (which assigns an individual DQN model for each zone in the building and trains them for 100 epochs, with one-month data for each epoch), DQN^* (standard deep Q learning method with m^n selections in the last layer [13]), and the direct training result of our method without transferring. Moreover, the DQN method is trained with 50, 100, and 150 training epochs (months), respectively, to show the impact of training time. As shown in the table, all learning-based methods demonstrate significant energy cost reduction over ON-OFF control. DQN^* shows slightly higher cost and violation rate, when compared to DQN after 150 epochs. Our approach with Algorithm 1 (i.e., not transferred) achieves the lowest violation rate among all learning-based methods, while providing a low cost.

Table 4 shows the **main comparison results** of our transfer learning approach and other baselines on four-zone building 2 and four-zone building 3. ON-OFF, DQN and DQN^* are directly trained on those two buildings. DQN_T^* is a transfer learning approach that transfers a well-trained DQN^* model on four-zone building 1 to the target building (four-zone building 2 or 3). Our approach transfers our trained four-zone building 1 model (last line in Table 3) to the target building. From Table 4, we can see that for both four-zone building 2 and 3, with 150 training epochs, DQN and DQN^* provide lower violation rate and cost than ON-OFF control, although DQN^* cannot meet the temperature violation requirement. And the other transfer learning approach DQN_T^* shows very high violation rate. In comparison, our approach achieves extremely low

temperature violation rate and a relatively low energy cost without any fine-tuning after transferring (i.e., EP is 0). We may fine tune the controller for 1 epoch (month) after transferring to further reduce the energy cost (i.e., EP is 1), at the expense of slightly higher violation rate (but still meeting the requirement). More studies on fine-tuning can be found in Section 4.5. Figure 3 (left) also shows the temperature over time for the target four-zone building 2, and we can see that it is kept well within the bounds.

4.3 Transfer from n-zone to m-zone

We also study the transfer from an n-zone building to an m-zone building. This is a difficult task because the input and output dimensions are different, presenting significant challenges for DRL network design. Here, we conduct experiments for transferring HVAC controller for four-zone building 1 to five-zone building 1 and seven-zone building 1, and the results are presented in Table 5. For these cases, DQN^* and DQN_T^* cannot provide feasible results as the m^n action space is too large for them, and the violation rate does not go down even after 150 training epochs. DQN [29] also leads to high violation rate. In comparison, our approach achieves both low violation rate and low energy cost. Figure 3 (middle and right) shows the temperature over time (kept well within the bounds) for the two target buildings after using our transfer approach.

4.4 Transfer from n-zone to n-zone with different HVAC equipment

In some cases, the target building may have different HVAC equipment (or a building may have its equipment upgraded). The new HVAC equipment may be more powerful or have a different number of control levels, making the original controller not as effective. In such cases, our transfer learning approach provides an effective solution. Here we conduct experiments on transferring our controller for the original HVAC equipment (denote as AC 1, which has two control levels and used in all other experiments) to the same building with new HVAC equipment (denoted as AC2, which has five control levels; and AC3, which has double max airflow rate and double air conditioner power compared to AC1). The experimental results are shown in Table 6. We can see that our approach provides zero violation rate after transferring, and the energy cost can be further reduced with the fine tuning process.

4.5 Fine-tuning study

After transferring, although our method has already gained a great performance without fine-tuning, further training is still worth considering because it may provide even lower energy cost. We record the change of cost and violation rate when fine-tuning our method transferred from four-zone building 1 to four-zone building 2. The results are shown in Figure 4.

4.6 Discussion

4.6.1 Transfer from n-zone to n-zone with different weather. As presented in [18], the Q-learning controller with weather that has a larger temperature range and variance is easy to be transferred into the environment with the weather that has a smaller temperature range and variance, but it is much harder in the opposite direction. This conclusion is similar to what we observed for our approach.
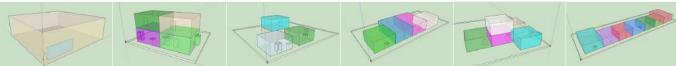
Figure 2: Different building models used in our experiments. From left to right, the models are one-zone building 1, four-zone building 1, four-zone building 2 , four-zone building 3, five-zone building 1, seven-zone building 1. Compared to four-zone building 1, four-zone building 2 has different layout and wall material; four-zone building 3 has different layout, wall material, and room size; five-zone building 1 has different number of zones, layout, and wall material; and seven-zone building 1 has different number of zones, layout, wall material, and room size.

Source building	Target building	θ_1	θ_2	θ_3	θ_4	μ_1	μ_2	μ_3	μ_4	☑	Cost
1-zone 1	*1-zone 1*	1.62%	-	-	-	1.11	-	-	-	✓	248.43
1-zone 1	*4-zone 2*	1.88%	9.43%	10.19%	14.07%	0.44	0.97	1.04	1.17	✗	308.13

Table 2: This table shows the experiment that transfers a single-zone DQN model (trained on one-zone building 1) to every zone of four-zone building 2. The high violation rate shows that such a straightforward scheme may not yield good results and more sophisticated methods such as ours are needed.

Method	Building	EP	θ_1	θ_2	θ_3	θ_4	μ_1	μ_2	μ_3	μ_4	☑	Cost
ON-OFF	*4-zone 1*	0	0.08%	0.08%	0.23%	0.19%	0.01	0.03	0.08	0.08	✓	329.56
DQN[29]	*4-zone 1*	50	1.21%	22.72%	9.47%	20.66%	0.68	2.46	1.61	2.07	✗	245.08
DQN[29]	*4-zone 1*	100	0.0%	0.53%	0.05%	0.93%	0.0	0.46	0.40	1.09	✓	292.91
DQN[29]	*4-zone 1*	150	0.0%	0.95%	0.03%	1.59%	0.0	0.52	0.17	1.17	✓	**278.32**
DQN*	*4-zone 1*	150	1.74%	2.81%	1.80%	2.76%	0.45	0.79	1.08	1.22	✓	289.09
Ours	*4-zone 1*	150	**0.0%**	**0.04%**	**0.0%**	**0.03%**	0.0	0.33	0.0	0.11	✓	297.42

Table 3: Results of different methods on four-zone building 1. Apart from the ON-OFF control, all others are the training results without transferring. The training model in the last row is used as the transfer model to other buildings in our method.

Method	Building	EP	θ_1	θ_2	θ_3	θ_4	μ_1	μ_2	μ_3	μ_4	☑	Cost
ON-OFF	*4-zone 2*	**0**	**0.0%**	**0.0%**	**0.0%**	**0.02%**	0.0	0.0	0.0	0.46	✓	373.78
DQN[29]	*4-zone 2*	50	0.83%	49.22%	46.75%	60.48%	0.74	2.93	3.18	3.39	✗	258.85
DQN[29]	*4-zone 2*	100	0.0%	1.67%	1.23%	3.58%	0.0	0.92	0.77	1.62	✓	352.13
DQN[29]	*4-zone 2*	150	0.0%	2.52%	1.67%	4.84%	0.0	1.64	1.56	1.61	✓	337.33
DQN*	*4-zone 2*	150	1.16%	2.71%	2.17%	6.44%	0.61	1.11	0.77	1.11	✗	323.72
DQN*$_T$	*4-zone 2*	0	12.35%	19.10%	10.39%	23.59%	2.47	4.67	2.27	5.22	✗	288.73
Ours	*4-zone 2*	**0**	**0.0%**	**0.0%**	**0.0%**	**0.07%**	0.0	0.0	0.0	0.88	✓	338.45
Ours	*4-zone 2*	1	0.09%	3.44%	1.91%	4.06%	0.33	1.04	0.96	1.35	✓	**297.03**
ON-OFF	*4-zone 3*	**0**	**0.0%**	**0.19%**	**0.0%**	**0.0%**	0.0	0.02	0.0	0.0	✓	360.74
DQN[29]	*4-zone 3*	50	0.68%	47.21%	44.61%	56.19%	0.74	3.15	2.92	3.60	✗	267.29
DQN[29]	*4-zone 3*	100	0.34%	2.53%	2.21%	5.59%	0.01	1.18	0.85	1.18	✗	342.08
DQN[29]	*4-zone 3*	150	0.0%	1.55%	1.68%	3.79%	0.0	1.09	1.18	1.51	✓	334.89
DQN*	*4-zone 3*	150	7.09%	13.85%	2.87%	2.16%	1.26	1.48	1.42	1.01	✗	316.93
DQN*$_T$	*4-zone 3*	0	13.31%	8.11%	3.18%	0.66%	1.25	3.48	2.27	0.69	✗	294.23
Ours	*4-zone 3*	**0**	**0.0%**	**0.28%**	**0.0%**	**0.0%**	0.0	0.37	0.0	0.0	✓	340.40
Ours	*4-zone 3*	1	0.23%	2.74%	0.04%	0.13%	0.34	1.73	0.12	0.31	✓	**331.47**

Table 4: Comparison between our approach and other baselines. The top half shows the performance of different controllers on four-zone building 2, including ON-OFF controller, DQN from [29] trained with different number of epochs, the standard Deep Q-learning method (DQN^*) and its transferred version from four-zone building 1 (DQN^*_T), and our approach transferred from four-zone building 1 (without fine-tuning and with 1 epoch tuning, respectively). We can see that our method achieves the lowest violation rate and very low energy cost after transferring without any further tuning/training. We may fine tune our controller with 1 epoch (month) of training and achieve the lowest cost, at the expense of slightly higher violation rate (but still meeting the requirement). The bottom half shows the similar comparison results for four-zone building 3.

We tested the weather from Riverside, Buffalo, and Los Angeles, which is shown in Figure 5. The results show that our approach can easily be transferred from large range and high variance weather (Riverside) to small range and low variance weather (Buffalo and Los Angeles(LA)), but not vice versa. Fortunately, the transferring for a new building is still not affected, because our approach can use the building models in the same region or obtain the weather data in that region and create a simulated model for transferring.

4.6.2 Different settings for ON-OFF control. Our back-end network (inverse building network) is learned from the dataset collected by an ON-OFF control with low temperature violation rate. In practice, it is flexible to determine the actual temperature boundaries for ON-OFF control. For instance, the operator may set the temperature
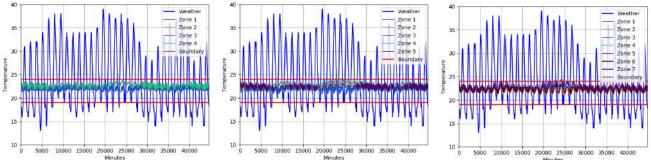
Figure 3: Temperature of four-zone building 2 (left), 5-zone building 1 (middle), and 7-zone building 1 (right) after transfer.

Method	Building	EP	Aθ	Mθ	Aμ	Mμ	☑	Cost
ON-OFF	5-zone 1	0	**0.45%**	**2.2%**	0.24	1.00	✓	373.90
DQN[29]	5-zone 1	50	38.65%	65.00%	2.60	3.81	✗	263.79
DQN[29]	5-zone 1	100	4.13%	11.59%	4.66	1.47	✗	326.50
DQN[29]	5-zone 1	150	2.86%	10.94%	0.89	1.63	✗	323.78
Ours	5-zone 1	0	0.47%	2.34%	0.33	1.42	✓	339.73
Ours	5-zone 1	1	2.41%	4.48%	1.02	1.64	✓	**323.26**
ON-OFF	7-zone 1	0	**0.37%**	**2.61%**	0.04	0.30	✓	392.56
DQN[29]	7-zone 1	50	28.14%	54.28%	2.76	3.06	✗	248.38
DQN[29]	7-zone 1	100	5.19%	18.91%	1.12	1.69	✗	277.87
DQN[29]	7-zone 1	150	4.48%	18.34%	1.22	1.98	✗	284.51
Ours	7-zone 1	0	0.42%	2.79%	0.10	0.43	✓	332.07
Ours	7-zone 1	1	**0.77%**	**1.16%**	0.77	1.21	✓	**329.81**

Table 5: Comparison of our approach and baselines on five-zone building 1 and seven-zone building 1.

Method	AC	EP	Aθ	Mθ	Aμ	Mμ	☑	Cost
ON-OFF	AC 2	0	**0.15%**	**0.23%**	0.05	0.08	✓	329.56
DQN[29]	AC 2	50	20.28%	35.56%	1.73	2.66	✗	229.41
DQN[29]	AC 2	100	1.25%	2.69%	0.61	1.20	✓	270.93
DQN[29]	AC 2	150	1.49%	2.87%	0.60	1.02	✓	263.92
Ours	AC 2	0	**0.0%**	**0.0%**	0.0	0.0	✓	303.37
Ours	AC 2	1	2.06%	4.20%	0.97	1.30	✓	**262.23**
ON-OFF	AC 3	0	**0.01%**	**0.05%**	0.22	0.88	✓	317.53
DQN[29]	AC 3	50	2.85%	3.76%	1.37	1.90	✓	321.03
DQN[29]	AC 3	100	0.69%	1.20%	0.53	0.99	✓	**265.46**
DQN[29]	AC 3	150	0.62%	1.07%	0.47	0.65	✓	266.86
Ours	AC 3	0	**0.0%**	**0.0%**	0.0	0.0	✓	316.16
Ours	AC 3	1	0.84%	1.42%	0.54	0.78	✓	269.24

Table 6: Comparison under different HVAC equipment.

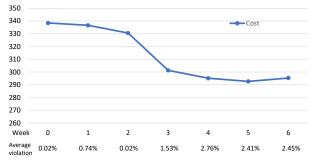| Week	0	1	2	3	4	5	6
Average violation	0.02%	0.74%	0.02%	1.53%	2.76%	2.41%	2.45%

Figure 4: Fine-tuning results of our approach for four-zone building 2. Our approach can significantly reduce energy cost after fine-tuning for 3 weeks, while keeping the temperature violation rate at a low level.
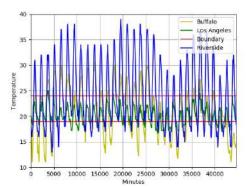

Figure 5: The visualization of different weathers. The yellow line is the Buffalo weather, the green line is the LA weather, the blue line is the Riverside weather, and the red lines are the comfortable temperature boundary.

Building	Source	Target	EP	Aθ	Mθ	☑	Cost
4-zone 1	LA	LA	150	0.68%	1.71%	✓	82.01
4-zone 1	Buffalo	Buffalo	150	0.64%	1.14%	✓	101.79
4-zone 1	Riverside	Riverside	150	0.02%	0.04%	✓	297.42
4-zone 1	Riverside	LA	0	0.0%	0.0%	✓	105.17
4-zone 1	Riverside	Buffalo	0	0.0%	0.0%	✓	134.28
4-zone 1	LA	Riverside	0	71.77%	89.34%	✗	158.06
4-zone 1	Buffalo	Riverside	0	54.92%	81.89%	✗	180.20

Table 7: Transferring between different weathers.

Method	Upper-Bound	EP	Aθ	Mθ	Cost
ON-OFF	23	0	0.01%	0.02%	373.78
ON-OFF	24	0	61.45%	73.69%	256.46
ON-OFF	25	0	98.56%	99.99%	208.79
Ours	23	0	0.02%	0.07%	338.45
Ours	24	0	0.02%	0.07%	338.08
Ours	25	0	0.02%	0.07%	338.08

Table 8: Results of testing using different boundary.

bound of ON-OFF control to be within the human comfortable temperature boundary (what we use for our method) or just the same as the human comfortable temperature boundary, or even a little out of boundary to save energy cost. Thus, we tested the performance of our method by collecting data under different ON-OFF boundary settings. Results in Table 8 shows that with different boundary settings, supervised learning can stably learn from building-specific behaviors.

5 CONCLUSION

In this paper, we present a novel transfer learning approach that decomposes the design of the neural network based HVAC controller into two sub-networks: a building-agnostic front-end network that can be directly transferred, and a building-specific back-end network that can be efficiently trained with offline supervise learning. Our approach successfully transfers the DRL-based building HVAC controller from source buildings to target buildings that can have a different number of thermal zones, different materials and layouts, different HVAC equipment, and even under different weather conditions in certain cases.

REFERENCES

[1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. 2019. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113* (2019).

[2] Enda Barrett and Stephen Linder. 2015. *Autonomous HVAC Control, A Reinforcement Learning Approach.* Springer.

[3] Yujiao Chen, Zheming Tong, Yang Zheng, Holly Samuelson, and Leslie Norford. 2020. Transfer learning with deep neural networks for model predictive control of HVAC and natural ventilation in smart buildings. *Journal of Cleaner Production* 254 (2020), 119866.

[4] Giuseppe Tommaso Costanzo, Sandro Iacovella, Frederik Ruelens, Tim Leurs, and Bert J Claessens. 2016. Experimental analysis of data-driven control for a building heating system. *Sustainable Energy, Grids and Networks* 6 (2016), 81–90.

[5] Drury B. Crawley, Curtis O. Pedersen, Linda K. Lawrie, and Frederick C. Winkelmann. 2000. EnergyPlus: Energy Simulation Program. *ASHRAE Journal* 42 (2000).

[6] Felipe Leno Da Silva and Anna Helena Reali Costa. 2019. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research* 64 (2019), 645–703.

[7] Pedro Fazenda, Kalyan Veeramachaneni, Pedro Lima, and Una-May O'Reilly. 2014. Using reinforcement learning to optimize occupant comfort and energy usage in HVAC systems. *Journal of Ambient Intelligence and Smart Environments* (2014), 675–690.

[8] Guanyu Gao, Jie Li, and Yonggang Wen. 2019. Energy-efficient thermal comfort control in smart buildings via deep reinforcement learning. *arXiv preprint arXiv:1901.04693* (2019).

[9] Guanyu Gao, Jie Li, and Yonggang Wen. 2020. DeepComfort: Energy-Efficient Thermal Comfort Control in Buildings via Reinforcement Learning. *IEEE Internet of Things Journal* (2020).

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. 6.5 Back-Propagation and Other Differentiation Algorithms. *Deep Learning* (2016), 200–220.

[11] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. 2017. Learning invariant feature spaces to transfer skills with reinforcement learning. *ICLR* (2017).

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision.* 1026–1034.

[13] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. 2018. Deep q-learning from demonstrations. In *AAAI.*

[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[15] Neil E Klepeis, William C Nelson, Wayne R Ott, John P Robinson, Andy M Tsang, Paul Switzer, Joseph V Behar, Stephen C Hern, and William H Engelmann. 2001. The National Human Activity Pattern Survey (NHAPS): a resource for assessing exposure to environmental pollutants. *Journal of Exposure Science & Environmental Epidemiology* 11, 3 (2001), 231–252.

[16] B. Li and L. Xia. 2015. A multi-grid reinforcement learning method for energy conservation and comfort of HVAC in buildings. *IEEE International Conference on Automation Science and Engineering (CASE)*, 444–449.

[17] Yuanlong Li, Yonggang Wen, Dacheng Tao, and Kyle Guan. 2019. Transforming cooling optimization for green data center via deep reinforcement learning. *IEEE*

[18] Paulo Lissa, Michael Schukat, and Enda Barrett. 2020. Transfer Learning Applied to Reinforcement Learning-Based HVAC Control. *SN Computer Science* 1 (2020).

[19] Y. Ma, F. Borrelli, B. Hencey, B. Coffey, S. Bengea, and P. Haves. 2012. Model Predictive Control for the Operation of Building Cooling Systems. *IEEE Transactions on Control Systems Technology* 20, 3 (2012), 796–803.

[20] Mehdi Maasoumy, Alessandro Pinto, and Alberto Sangiovanni-Vincentelli. 2011. Model-based hierarchical optimal control design for HVAC systems. In *Dynamic Systems and Control Conference*, Vol. 54754. 271–278.

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

[22] Mehdi Maasoumy, Meysam Razmara, Mahdi Shahbakhti, and Alberto Sangiovanni Vincentelli. 2014. Selecting building predictive control based on model uncertainty. In *2014 American Control Conference*. IEEE, 404–411.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[24] Aviek Naug, Ibrahim Ahmed, and Gautam Biswas. 2019. Online energy management in commercial buildings using deep reinforcement learning. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 249–257.

[25] D Nikovski, J Xu, and M Nonaka. 2013. A method for computing optimal set-point schedules for HVAC systems. In *REHVA World Congress CLIMA*.

[26] U.S. Department of Energy. 2011. *Buildings energy data book.*

[27] Saran Salakij, Na Yu, Samuel Paolucci, and Panos Antsaklis. 2016. Model-Based Predictive Control for building energy management. I: Energy modeling and optimal control. *Energy and Buildings* 133 (2016), 345–358.

[28] T. Wei, S. Ren, and Q. Zhu. 2019. Deep Reinforcement Learning for Joint Datacenter and HVAC Load Control in Distributed Mixed-Use Buildings. *IEEE Transactions on Sustainable Computing* (2019), 1–1.

[29] Tianshu Wei, Yanzhi Wang, and Qi Zhu. 2017. Deep reinforcement learning for building HVAC control. In *Proceedings of the 54th Annual Design Automation Conference 2017.* 1–6.

[30] Tianshu Wei, Qi Zhu, and Nanpeng Yu. 2015. Proactive demand participation of smart buildings in smart grid. *IEEE Trans. Comput.* 65, 5 (2015), 1392–1406.

[31] Michael Wetter. 2011. Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed. *Journal of Building Performance Simulation* 4, 3 (2011), 185–203.

[32] Stephen Wilcox and William Marion. 2008. Users manual for TMY3 data sets. (2008).

[33] Yu Yang, Seshadhri Srinivasan, Guoqiang Hu, and Costas J Spanos. 2020. Distributed Control of Multi-zone HVAC Systems Considering Indoor Air Quality. *arXiv preprint arXiv:2003.08208* (2020).

[34] Liang Yu, Yi Sun, Zhanbo Xu, Chao Shen, Dong Yue, Tao Jiang, and Xiaohong Guan. 2020. Multi-Agent Deep Reinforcement Learning for HVAC Control in Commercial Buildings. *IEEE Transactions on Smart Grid* (2020).

[35] Yusen Zhan and Mattew E Taylor. 2015. Online transfer learning in reinforcement learning domains. In *2015 AAAI Fall Symposium Series.*

[36] Zhiang Zhang, Adrian Chong, Yuqi Pan, Chenlu Zhang, Siliang Lu, and Khee Poh Lam. 2018. A deep reinforcement learning approach to using whole building energy model for hvac optimal control. In *2018 Building Performance Analysis Conference and SimBuild*, Vol. 3. 22–23.

[37] Zhiang Zhang and Khee Poh Lam. 2018. Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system. In *Proceedings of the 5th Conference on Systems for Built Environments.* 148–157.