

Learning-based Framework for Sensor Fault-Tolerant Building HVAC Control with Model-assisted Learning

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

Yangyang Fu
Texas A&M University
College Station, USA
yangyang.fu@tamu.edu

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

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

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

ABSTRACT

As people spend up to 87% of their time indoors, intelligent Heating, Ventilation, and Air Conditioning (HVAC) systems in buildings are essential for maintaining occupant comfort and reducing energy consumption. These HVAC systems in smart buildings rely on real-time sensor readings, which in practice often suffer from various faults and could also be vulnerable to malicious attacks. Such faulty sensor inputs may lead to the violation of indoor environment requirements (e.g., temperature, humidity, etc.) and the increase of energy consumption. While many model-based approaches have been proposed in the literature for building HVAC control, it is costly to develop accurate physical models for ensuring their performance and even more challenging to address the impact of sensor faults. In this work, we present a novel learning-based framework for sensor fault-tolerant HVAC control, which includes three deep learning based components for 1) generating temperature proposals with the consideration of possible sensor faults, 2) selecting one of the proposals based on the assessment of their accuracy, and 3) applying reinforcement learning with the selected temperature proposal. Moreover, to address the challenge of training data insufficiency in building-related tasks, we propose a model-assisted learning method leveraging an abstract model of building physical dynamics. Through extensive experiments, we demonstrate that the proposed fault-tolerant HVAC control framework can significantly reduce building temperature violations under a variety of sensor fault patterns while maintaining energy efficiency.

CCS CONCEPTS

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

BuildSys '21, November 17–18, 2021, Coimbra, Portugal

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-9114-6/21/11...\$15.00

<https://doi.org/10.1145/3486611.3486644>

KEYWORDS

HVAC control, Sensor fault-tolerant, Deep learning

ACM Reference Format:

Shichao Xu, Yangyang Fu, Yixuan Wang, Zheng O'Neill, and Qi Zhu. 2021. Learning-based Framework for Sensor Fault-Tolerant Building HVAC Control with Model-assisted Learning. In *The 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '21)*, November 17–18, 2021, Coimbra, Portugal. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3486611.3486644>

1 INTRODUCTION

People spend up to 87% of their time in enclosed buildings nowadays [16]. As Heating, Ventilation, and Air-Conditioning (HVAC) systems control the indoor environment of buildings and have a significant impact on occupant comfort, productivity, and physical/mental health, it is important to ensure their performance and reliability. In these systems, sensors, in particular temperature sensors, play a vital role in collecting real-time environment condition and facilitating HVAC applications. However, temperature sensors are not always in normal working condition, due to passive faults and active cyber-attacks. Passive sensor faults such as sensor bias and sensor drifting over a long time contribute more than 25% to the variable air volume (VAV) terminal unit faults [28]. Cyber-attacks on HVAC control systems (i.e., corruption of temperature sensor readings to affect critical control programs) are becoming possible due to increasing connectivity of buildings to external networks for supporting remote management and cloud-based analytics. For example, Building Automation and Control Networks (BACnet) [24], the most popular communication protocol for buildings, has been reported to have multiple vulnerabilities that can be used to launch cyber-attacks on building control systems [11]. Moreover, HVAC systems still need to provide services when under faults or attacks, as diagnosing the problems and fixing the sensors often takes a significant amount of time. This highlights the increasing need for developing HVAC controls that can tolerate sensor faults and cyber-attacks and increase system resilience.

There are a number of works in the literature related to sensor fault-tolerant control for building energy systems. Ma and Wang [20] proposed a fault-tolerant model predictive control strategy to provide resilient operation of a building chiller plant system under typical faults such as condenser water supply temperature sensor bias. Yang et al. [40] presented an online fault-tolerant

control strategy for fixed bias faults in the supply air temperature sensor. The sensor faults are detected by using a pre-trained support vector regression (SVR) model. [9] employed a rule-based method (e.g., using sensor reading from the nearest zone) to mitigate the zone air temperature sensor reading spikes. The work in [26] built a physical model for a multi-zone building and with zone air temperature sensor faults, and assumed that only one thermal zone would be affected by the sensor fault at a time. Faults in sensors other than temperature sensors are also studied for tolerant control design. Wang et. al. [33] applied a neural network model to detect and compensate outdoor air flow rate sensor faults, and provided a fault-tolerant control strategy to regain the control of outdoor air flow rate. However, the above literature has the following limitations: 1) simple assumptions in terms of fault occurrences are used: for instance, [26] assumed that only one thermal zone would be affected by the zone air temperature sensor fault at a time, which is often not the case in practice; 2) studies were mostly designed for passive faults such as fixed sensor bias [12, 20, 40], and might not be applied to active attacks that only last for a short duration but with high intensity; 3) significant efforts are required to obtain an accurate online state predictor, such as detailed physics-based models or SVR model, for fault detection in the fault-tolerant control. Therefore, how to provide resilient control for HVAC systems under abnormal sensor readings still remains an open challenge.

In this work, we develop a *learning-based sensor fault-tolerant control framework* for building HVAC systems with novel deep neural network-based learning techniques. Specifically, our framework includes three major components. First, as the raw sensor readings of the indoor temperature may be faulty, a neural network-based temperature predictor is designed based on historical sensor data to provide an alternative estimation of the true temperature. Then, both proposals of the indoor temperature (raw sensor reading and the temperature predictor output) are sent to a neural network-based selector, which assesses the two temperature proposals with consideration of the historical trend and selects one deemed more trustworthy. Finally, a deep reinforcement learning (DRL) based HVAC controller takes the chosen temperature as the current system state and applies control actuation. These learning-based techniques together provide a robust HVAC control framework that can maintain desired temperature and reduce energy consumption under sensor faults.

While our machine learning based techniques can remove the need for developing detailed and costly building physical models, they face their own challenges in training data availability. In particular, for a new building, we may have to wait for months to collect enough data for training the learning-based components. To address this challenge, we propose a *model-assisted learning* approach that helps the learning components extract knowledge from an *abstract physical model* and only requires a limited amount of additional labeled data collected from real buildings for training. There are a number of abstract physical models available in the literature [21, 32]. They require much less effort to develop than the accurate physical models (e.g., those used in EnergyPlus [3]). While they alone are often not accurate enough for building HVAC control, their capturing of the underlying physical laws can guide the learning process for the neural network-based components and significantly improve the learning effectiveness.

To summarize, our work makes the following contributions:

- We present a novel sensor fault-tolerant learning-based framework to achieve sensor fault resilience on building HVAC control. The framework includes three neural network-based components: a temperature predictor that estimates the true indoor temperature, a selector that assesses the predictor output and the raw sensor reading and selects one, and a DRL-based controller that generates the control signal.
- We develop a novel learning method called model-assisted learning, which leverages the knowledge from an abstract physical model to enable learning with a small amount of labeled data.
- We conduct a number of experiments on buildings with a single thermal zone and multiple zones, and demonstrate the effectiveness of our fault-tolerant framework under various types of sensor anomalies. We also highlight how model-assisted learning can improve the learning process and reduce the need for training data.

The rest of the paper is as follows. Section 2 discusses further about the related literature. Section 3 introduces our approach, including the design of the sensor fault-tolerant framework and model-assisted learning. Section 4 shows the experiments and related ablation studies. Section 5 concludes the paper.

2 RELATED WORK

2.1 Building HVAC control

Building HVAC supervisory controllers can be categorized into two groups, *model-based controllers* and *model-free controllers*. Classic model-based HVAC controllers are often developed based on fundamental physics laws (e.g., considering heat transfer and air-flow balance). For example, [21] designed a hierarchical control algorithm based on modeling building thermal dynamics as an RC network, which uses resistance and capacitance elements to model the building envelope heat transfer. [32] also used an RC network model and designed a model predictive control algorithm for minimizing the building energy consumption. There are other works [30, 38] that use similar abstract physical models. However, while being easy to develop and fast to run, these abstract physical models often suffer from inaccuracy. In contrast, detailed physical models such as EnergyPlus consider a variety of complex factors, including building layout, wall materials, light, shading, occupant behaviors, etc. They are much more accurate, but are typically hard to build and slow to run.

Model-free HVAC controllers usually learn control strategies from historical data. In recent years, DRL-based methods have been explored in works such as [35, 41], where techniques such as deep Q-learning (DQN) and asynchronous advantage actor-critic algorithms (A3C) are applied. Methods have also been proposed to learn DRL parameters by leveraging building simulation tools [8, 17, 23, 39]. In this paper, we combine the strength of both model-free and model-based methods, by developing a learning-based framework with neural network-based components and leveraging abstract physical models to improve the learning process.

2.2 Addressing sensor faults in buildings

There has been a number of works in the literature addressing sensor faults in buildings. In [5], a fault detection method based on correlation analysis was proposed for detecting sensor bias or complete failure. [6] proposed a neural network-based strategy with clustering analysis to detect sensor faults in the HVAC system and diagnose the sources. [34] presented an online strategy based on the principal component analysis (PCA) to detect, diagnose and validate sensor faults in centrifugal chillers. More investigations can be found in [4, 7, 13, 19, 22, 29]. However, these works focus on fault detection and diagnosis, not fault-tolerant control.

There are some existing works for sensor fault-tolerant control in building energy systems, such as [9, 12, 20, 26, 33, 40]. For instance, Gunes et. al. [9] followed the model-based design paradigm and used rule-based methods to mitigate the negative effect of specific sensor faults. Papadopoulos et. al. [26] built a complex physical model for building, and designed a fault model based on the assumption that sensor faults occur in a single zone at each time. Jin and Du [12] used principal component analysis, joint angle method and compensatory reconstruction to detect, isolate and reconstruct the fixed bias fault in supply air temperature sensors. However, as we outlined in the introduction, the above studies have significant limitations in the usage of simple or restricted assumptions, the focus on only passive faults with fixed sensor bias, and the need of significant efforts for obtaining an accurate online state predictor (e.g., with detailed physics-based models or SVR model). In contrast, our learning-based approach provides resilient control in broader and more practical cases.

2.3 Learning with limited data and abstract physical model

When dealing with a limited amount of labeled data in training, techniques such as weakly supervised learning [31, 42] and semi-supervised learning [1, 27, 43] are often considered. However, in our case, even obtaining unlabeled data from real building operations could be a long process. Thus, we leverage the information from abstract physical models such as those in [21, 32] to reduce the data needed for training. This approach is in principle related to model distillation techniques [10, 14] that distill the physical model into a neural network and then fine-tune the network with available labeled data. However, unlike in the case for those approaches (which focus on domains such as computer vision), there is not enough unlabeled data in the realistic data distribution that can be fed into the model for distillation in our problem. Thus, we propose model-assisted learning to overcome this difficulty, by leveraging abstract physical models to generate better initial points for model fine-tuning.

3 METHODOLOGY

3.1 System model

We adopted a multi-zone building model with the fan-coil system from [35, 39], where there is a building with n thermal zones, and a fan-coil system is equipped to provide the conditioned air at a given supply air temperature T^{air} for each thermal zone. The airflow rate in each zone is chosen from multiple discrete levels

$\{f_1, f_2, \dots, f_m\}$, and corresponding to m control actions a_i for each zone i . With all n thermal zones, the control action set is denoted as $A = \{a_1, a_2, \dots, a_n\}$. In this paper, we denote the current physical time as t , the ambient temperature, indoor temperature for zone i , and the control action at time t as T_t^{out} , $T_t^{in(i)}$, A_t , respectively, and we set $T_t^{in} = \{T_t^{in(i)} | i \in 1 \dots n\}$. The system sends current states (indoor and ambient temperatures) to the HVAC system with a period of Δt_s (which is the simulation period on building simulation platform), and the building HVAC controller provides the control signal (supply airflow rates) with a period of Δt_c (i.e., the control period).

3.2 Sensor fault-tolerant DRL framework

Fig. 1 depicts the overview of our sensor fault-tolerant DRL framework. It includes three parts: the first part on the left is a neural network-based temperature predictor for providing an alternative estimation (rather than the raw sensor reading) of the indoor temperature, the second part in the middle is a proposal selector that assesses the temperature proposals from the raw sensor reading T_t^{in} and the temperature prediction T_t^{pre} and selects one, and the third part on the right is a DRL-based HVAC controller. With the design of the predictor and the selector, the DRL controller receives a refined indoor temperature reading as part of its inputs and can maintain a stable performance against sensor faults or attacks. The details of each module are introduced in the following sections. Compared with using a single neural network to generate an estimated temperature for controller input, our predictor and selector design enables us to leverage the raw sensor readings when they are not faulty, which is especially useful when the temperature predictor does not provide accurate estimations because of training data availability (and as we will introduce in the later sections, the selector has more augmented training data than the predictor to ensure its accuracy). Moreover, note that all the modules are trained individually and assembled into the framework after training.

3.2.1 Temperature predictor. The temperature predictor aims to provide an indoor temperature prediction for the current step based on the historical sensor readings with possible faults and other system states. Note that we mark the current system state as S_t , where $S_t = (t, T_t^{in}, T_t^{out})$.

Firstly, the temperature predictor is a neural network that consists of five fully-connected layers. Except for the last layer, all layers are filtered by a ReLU activation function, and all fully-connected layers are sequentially connected (detailed neuron number settings can be found later in Table 1 of Section 4). In the test stage of the temperature predictor, the network takes the historical states aligned with the historical control actions (airflow rate) as the data inputs at time t , and then outputs a current indoor temperature prediction value T_t^{pre} .

The training data for the predictor network is collected by running a straightforward ON-OFF controller on the building HVAC system for several days (in experiments we use 8 days). For new buildings, this could be done during the first several days of their operation, in which case we may assume that the data collected over this short time period has not been polluted by sensors faults or attacks. We get a (state, action) sequence from (S_1, A_1) to (S_L, A_L) . For

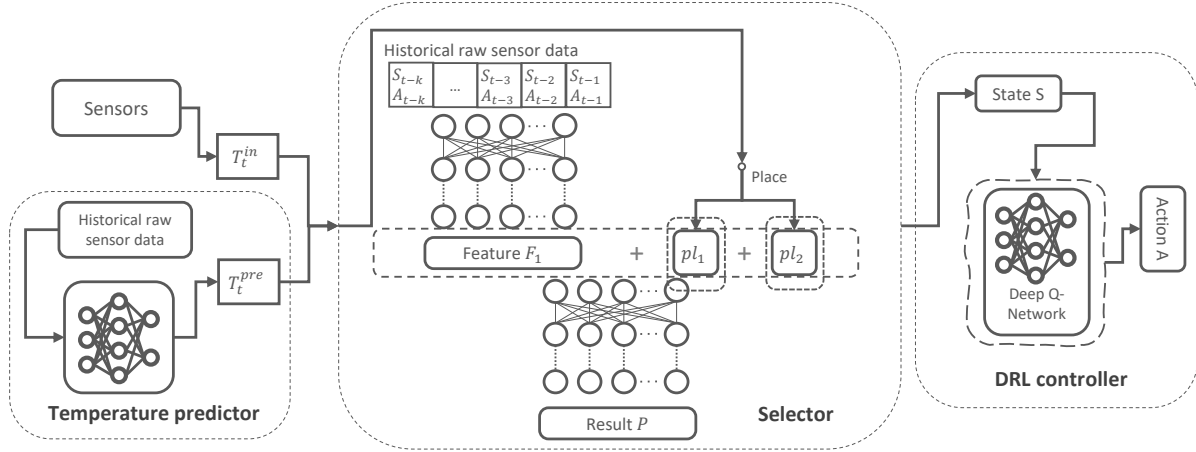


Figure 1: Overview of our sensor fault-tolerant framework for building HVAC system. There are three main components: two modules providing indoor temperature proposals on the left, a selector in the middle, a DQN-based HVAC controller on the right. The temperature proposals consist of the raw sensor reading T_t^{in} and the current temperature prediction T_t^{pre} that comes from the learned temperature predictor, which leverages the historical sensor data. The proposal selector provides a classification result to choose between the predictor output and the raw sensor value. Then, the DRL controller takes the selected indoor temperature proposal and calculates the corresponding control action.

the convenience of supervised training, we select data sequences

$$\{ \langle (S_{t-k}, A_{t-k}), (S_{t-k+1}, A_{t-k+1}), \dots, (S_{t-1}, A_{t-1}) \rangle \}$$

with length k and $t \in [k+1, L]$ from the historical data. These sequences are chosen with an interval v , which means that $t \in [k+1, L]$ is selected in the format $k+1, k+v+1, k+2v+1, \dots$. The collected data set is used as the training data inputs of the neural network, with the corresponding label S_t for each data sequence. Then, we train the neural network based on the loss function

$$\mathcal{L}_{pre} = \| (T_t^{pre} + T_{ofs}^{pre}) - T_t^{in} \|^2, \quad (1)$$

where T_t^{pre} is the temperature prediction at time t from the network's output, T_{ofs}^{pre} is an estimated offset for bringing the absolute mean value of the neural network's output close to zero, which lowers the difficulty for the neural network learning through the given data sequences (it is a fixed hyper-parameter; setting can be found in Table 1 later). T_t^{in} is the actual indoor temperature, which is the ground truth label. After finishing training, the predictor can take the historical system states containing the raw sensor reading to generate the temperature prediction. We should mention that these historical system states in the test stage may contain faulty sensor readings, so we also include some faulty sensor reading in the training data for temperature prediction. The designing of this training strategy using historical data with slight faults is inspired by our preliminary experiments, which indicated that adding slightly faulty sensor reading to the training data could increase the performance on temperature predictions, compared to training with non-faulty data or data with high frequency faulty data. In other words, to enhance the robustness of the temperature prediction, in the historical system states, we utilize the historical indoor temperature under independent and identically distributed (IID) faults with occurring probability P_{pre} . Using IID faults here means that a fault can happen at each individual simulation step with the

probability P_{pre} . If a fault occurs, a random number is uniformly selected from $[T_l^{out}, T_u^{out}]$, which is the upper and lower boundary of the ambient temperature, to replace the original indoor sensor temperature reading. The training of the temperature predictor benefits from such randomized faults, which results in more robust output of the trained predictor. Note that we set P_{pre} to a small value during training because the neural network's ability in tolerating the input noise is limited. While some small noises in the training data may enhance the network robustness, larger noises may negatively impact the training, making it harder to converge and reducing the overall performance.

3.2.2 Temperature proposal selector. The temperature proposal selector aims to choose the best candidate from the indoor temperature proposals and send it to the DRL controller for further control steps. We train this module in a self-supervised way, where all the training labels are generated automatically and the objective is to distinguish between the normal data and the faulty data. Apart from the comparison between the normal and faulty, we also make the comparison among the faulty data and indicate which one is closer to the actual temperature value. This extra comparison further boosts the proposal selector and helps it address the scenarios with inaccurate temperature proposals.

The temperature proposal selector module is made of a neural network that consists of eight layers. The selector firstly takes the historical system state and the historical control actions $\langle (S_{t-k}, A_{t-k}), (S_{t-k+1}, A_{t-k+1}), \dots, (S_{t-1}, A_{t-1}) \rangle$ as the part of the network input. Then this historical information will be sent to the first network layer. Including the first layer, there are four sequentially connected one-dimensional convolutional layers with the ReLU activation function on the bottom of the network. The output feature of these layers is two-dimensional in each data sample, and we convert it to a one-dimension feature vector F_1 . Then the rest of the network inputs are two selected indoor temperature proposals,

the raw sensor reading pl_1 and the temperature prediction value pl_2 , and they will be concatenated with the feature vector F_1 . One major motivation for such network design is that the historical states are sequential data with temporal locality. We thus choose one-dimensional convolutional layers to extract their feature. The temperature proposals are different type of data, and we concatenate the previous feature with these temperature proposals and fuse them using fully-connected layers. As shown in Fig. 1, four fully-connected layers receive features vector F_1 and with those two selected temperature proposals pl_1, pl_2 (note that the first three of them have ReLU activation function). The last fully-connected layer has two neurons, which will be sent to a softmax layer and output a binary classification result by selecting the index with the maximum output value.

Furthermore, the construction of the training data used for the temperature proposal selector differs from the previous module. The historical system state $S_{t-i} (i \in [1, k])$ and the historical control actions $A_{t-i} (i \in [1, k])$ are selected from the simulation data which is the same as in Section 3.2.1. The data in the two indoor temperature proposals contain both normal and faulty data. So the training data consists of three types:

- Training data: $\langle$ historical system states S_{t-i} , control actions A_{t-i} , ($i \in [1, k]$), normal temperature, faulty temperature $\rangle$. Label: (1, 0).
- Training data: $\langle$ historical system states S_{t-i} , control actions A_{t-i} , ($i \in [1, k]$), faulty temperature, normal temperature $\rangle$. Label: (0, 1).
- Training data: $\langle$ historical system states S_{t-i} , control actions A_{t-i} , ($i \in [1, k]$), faulty temperature, faulty temperature $\rangle$. Label: 1 is assigned to the value that is closer to the normal temperature. The other is assigned with 0.

Similar to the data construction strategy in the temperature predictor module, the historical system states we utilize include the faulty sensor readings. Specifically, for enhancing the robustness of the temperature proposal selector, we use the historical system states under the independent and identically distributed (IID) faults with occurring probability P_{sel} . Besides, during constructing these data-label pairs, we sample the faulty temperature three times for each normal temperature value in the first and second kind of data-label pair. For the last kind of data-label pair, we sample the faulty temperature data four times for each historical sequence. All faulty temperature readings come from the IID faults. Finally, we learn the temperature proposal selector network through the cross-entropy loss function. The learning rate lr_{sel} and training epochs l_{sel} are set as in Table 1 later.

3.2.3 DRL-based controller for building HVAC system. Because the thermal zone temperature in the next time step only relies on the current system state, the building HVAC control can be treated as a Markov decision process. We use a DQN-based DRL method that takes the current state S_t^{DRL} as inputs, which contain

- Current physical time t ,
- Current indoor air temperature T_t^{in} ,
- Current ambient air temperature T_t^{out} ,
- Current solar irradiance intensity Sun_t ,
- Weather forecast in the next three time steps.

The weather forecast includes ambient temperature and solar irradiance intensity $T_{t+1}^{out}, \dots, T_{t+3}^{out}, Sun_{t+1}, \dots, Sun_{t+3}$, which helps the network capture the trend of the environment. The deep Q-network Q provides the Q-value estimation of current control actions. The algorithm takes the control action with the maximum Q-value and sends it to the HVAC system.

Furthermore, the goal of this DRL controller is to minimize total energy cost while maintaining indoor temperature within a comfort temperature bound $[T_l, T_u]$. The reward function R_t collected from the control steps is designed accordingly as

$$R_t = \alpha \cdot R_c + \beta \cdot R_v \quad (2)$$

$$R_c = -cost(t-1, A_{t-1}) \quad (3)$$

$$R_v = -\sum_{i=1}^n \max(T_l - T_t^{in(i)}, 0) + \max(T_t^{in(i)} - T_u, 0) \quad (4)$$

where α and β are the scaling factors. R_c is the reward of energy cost, R_v is the reward of temperature violation with respect to comfort temperature bound $[T_l, T_u]$. $cost(t-1, A_{t-1})$ is a price function that gives the money cost of the HVAC system from control time $t-1$ to t under control action A_{t-1} . It is designed based on the local electricity price. Following the definition of the reward function, the update of deep Q-network is defined as

$$Q_{t+1}(S_t^{DRL}, A_t) = Q_t(S_t^{DRL}, A_t) + \eta_0(R_{t+1} + \gamma \max_{A_{t+1}} Q_t(S_{t+1}^{DRL}, A_{t+1}) - Q_t(S_t^{DRL}, A_t)) \quad (5)$$

where η_0 is the learning rate for the deep Q-network, and γ is the decay factor of the accumulative reward.

3.3 Model-assisted learning

Our sensor fault-tolerant framework has three modules that require neural network training. The performance of a learning model is typically strongly correlated with the amount of available labeled data. However, collecting labeled data from real building operations takes significant amount of time, which often leads to the problem of training data insufficiency. With the techniques in [2, 18], the training time and the required data of the DRL control module can be substantially reduced. With the special training data construction strategy introduced in Section 3.2.2, the selector also has sufficient data for training. Thus, we focus our effort on the possible data insufficiency issue for the temperature predictor. We develop a novel model-assisted learning method to combine a limited number of accurately-labeled data D^L with the knowledge we can gain from an abstract physical model M for the training, as shown in Fig. 2.

Abstract physical model: Here we introduce the abstract physical model that is used for model-assisted learning. The mass and energy conservation law for a building zone is presented in Equation (6), where the left of the equation represents energy changes in the zone, the first term at the right represents the introduced HVAC energy to the zone, and the second term at the right is the thermal load in the zone. The thermal load $\dot{q}_l$ is related to many building system and control parameters such as envelope constructions, internal heat gains, zone air temperature setpoints, etc., which eventually leads to a nonlinear differential equation to solve. For simplification, an abstract model for the zone air temperature dynamics is derived as

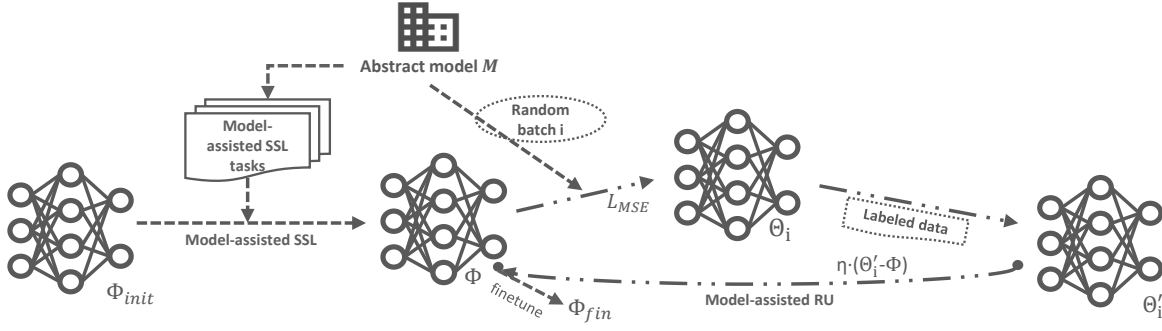


Figure 2: Overview of our model-assisted learning for training with a limited amount of labeled data and an abstract physical model, where the algorithm consists of two stages – model-assisted self-supervised learning (model-assisted SSL) and model-assisted redirected updating (model-assisted RU). The former stage creates auxiliary learning tasks from the abstract model, and the latter stage extracts knowledge leveraging the random batch from the physical model and explores a better updating direction. Then we get the final model through fine-tuning based on the pre-trained model from the previous two stages.

in Equation (7). This model explicitly relates zone air temperature to system thermal inertia (e.g., historical zone air temperatures), zonal supply air mass flowrate $\dot{m}$, outdoor air temperature T^{out} and estimated modeling error term e . $\hat{T}$ and T are the predicted and measured temperature, respectively. m is the zone air thermal mass. $\dot{m}$ is the zonal supply air mass flowrate. C_p is the zone air specific heat. e represents an error term. Superscripts *sa*, and *out* are the supply air, and outdoor air, respectively. α , β , and γ are identified coefficients observed from the given short-term historical data.

$$mC_p \frac{dT}{dt} = \dot{m}C_p(T^{sa} - T) + \dot{q}^l \quad (6)$$

$$\hat{T}_{t+1} = \alpha T_t + \beta \hat{m}_{t+1} + \gamma \hat{T}_{t+1}^{out} + e_{t+1} \quad (7)$$

$$e_{t+1} = \sum_{j=0}^{L-1} \frac{\hat{T}_{t-j} - T_{t-j}}{L} \quad (8)$$

Model-assisted learning algorithm: Our model-assisted learning consists of two stages: model-assisted self-supervised learning (called model-assisted SSL) and model-assisted redirected updating (called model-assisted RU). To begin with, we realize that the biggest challenge in this learning scenario is that we do not have enough training data (even unlabeled data), which makes the typical semi-supervised or weakly supervised learning methods not applicable. However, one available resource that we can leverage is the human-designed abstract physical models for buildings. While they may not accurately describe the building dynamics, they do reflect some of the fundamental physical laws for the system. By ‘extracting’ these physical laws, we can significantly improve the learning process and reduce the need for training data. The main idea is to utilize the abstract physical model for generating sampled states sequences in both the model-assisted SSL stage and the model-assisted RU stage. We will explain this in details below.

For each element u in the neural network input s , we can define its range based on its physical meaning. Then considering the range for all the elements in s , we can define a space H that contains all s in its range combinations and $s \in H$. Note that H is a space that is much larger than the actual data distribution for network inputs, which means that many unrealistic cases that will never happen in the real world might still occur when sampling from H .

In model-assisted learning process, a required step is to collect enough samples from data space H . However, we notice that the input size of the neural network (temperature predictor), $(2 + 2n)k$, is large. Taking $n = 4, k = 20$ for example, the sampling is on a 200-dimensional continuous data space, which is too expensive for simple uniform sampling. Thus, we only sample the first historical state uniformly among that sub-space of size $2 + 2n$, and then feed that historical state to the physical model M to predict the next historical state. Then we generate the latter historical states by repetitively applying the previous historical states to the physical model. In this way, we can collect the sample sequences of length k and form an input data set D . We then divide D into mini-batches and call them random batches $\{x|x \subset D\}$, and we denote the batch size of x as b . With the random batch, we can design the steps in model-assisted SSL and model-assisted RU.

In the first stage of model-assisted SSL, we aim to construct auxiliary learning tasks from the abstract model M to decide an pre-trained weights for the neural network. The sampled data $d \in x \subset D$ is a simulated states sequence based on the abstract physical model M , and the time length of the sequence is k . And we create k auxiliary learning tasks based on the input sequence d . Specifically, for auxiliary task i , it is a regression task. The corresponding training data is $\{(d_i, y_i)|d_i \text{ equals to } d \text{ except that the indoor temperature in } d \text{ at time step } i \text{ is set to } -1, y_i \text{ is the value of indoor temperature in } d \text{ at time step } i, d \in x \subset D\}$. In other word, we try to predict the missing state generated by the abstract model. The training step last for l_{MS_i} epochs with batch size as b_{MS} and learning rate as η_0 . In auxiliary task i , we also need to edit the original neural network with some changes. We keep the first three fully-connected layers but add two extra fully-connected layers (individually for each task i) following the third layer. The newly added layers will provide the output for task i . This means that we share the feature extraction layers among all the auxiliary learning tasks, and those tasks will help the neural network leverage the relation of variables in the states sequence for constructing pre-trained weights. The model-assisted SSL will be conducted for l_{MS} epochs, and we start from the randomly initialized neural network weights Φ_{init} . The auxiliary learning tasks are run in order from tasks 1 to k in each epoch, and then we get a pre-trained weight Φ for our next stage.

In the second stage of model-assisted RU, we target on redirecting the updating direction when extracting the knowledge from the abstract model. In each update step i , we start from the current network weights Φ_i (the initial weights in this stage is $\Phi_0 = \Phi$), and select a random batch $\mathbf{x}$ and apply the abstract model M on them to get the corresponding labels $\mathbf{y}$. Next, we are able to get a new model Θ_i by updating the parameters on Φ_i using the random batch $\mathbf{x}$ and its corresponding labels $\mathbf{y}$, which follows the equation

$$\Theta_i = \Phi_i - \eta_2 \nabla_{\Phi_i} \mathcal{L}_{MSE}(\Phi_i), \quad (9)$$

where $\mathcal{L}_{MSE}$ is the mean square error loss and η_2 is the learning rate. The training lasts for n_{iter} iterations, and uses a new sampling data batch for each iteration.

Next, we employ accurately labeled data D^L to further fine-tune the model Θ_i from the last step by l_{ft} epochs, and update to the model weights Θ'_i , as described in the following equation

$$\Theta'_i = \Theta_i - \eta_3 \nabla_{\Theta_i} \mathcal{L}_{target}(\Theta_i), \quad (10)$$

where $\mathcal{L}_{target}$ is the loss function for the target task and η_3 is the learning rate for this step.

Looking back to what we have done in this stage, we first use the random batch $\mathbf{x}$ to distill the physical model M as a further pre-trained model for the current step, and then we fine-tune the model using the accurately labeled data. The final performance of model Θ'_i should reflect the quality of the initial update from Φ_i to Θ_i , which depends on the corresponding random batch $\mathbf{x}$ and the abstract model M 's output knowledge $\mathbf{y}$. $\mathcal{L}_{target}$ shows a reference value considering the improvement brought by the Equation (9), while $\Theta'_i - \Phi_i$ provides a better updating direction for the current knowledge extraction step compared to the Equation (9). Thus, we determine the true updating step for the initial model Φ_i as

$$\Phi_i = \Phi_i - \eta_1 (\Theta'_i - \Phi_i) \quad (11)$$

Following this updating steps for n_{iter} iterations, we then use all the accurately labeled data D^L to fine-tune the extracted model Φ_i to achieve our target model Φ_{fin} . The fine-tuning step has the learning rate η_3 by l_{ft} epochs.

4 EXPERIMENTAL RESULTS

4.1 Fault patterns and metrics

Fault patterns: We consider two types of fault patterns for the indoor sensors in every thermal zone in our experiments. Both patterns could be caused by passive faults or cyber attacks.

- In the first type of faulty sensor readings, we postulate that the fault happens at each time step with a probability p_1 . Note that the fault can happen in each simulation step, not only on the control steps. If the fault occurs, it uniformly selects a random number from $[T_l^{out}, T_u^{out}]$ (which is the upper and lower boundary of the ambient temperature in our experiments) to replace the original sensor temperature reading. We call this type of faults the IID faults because they have the same probability, same distribution, and independent at each time step.
- For the second type of faulty sensor reading, the fault happens at each time step with a probability p_2 . The difference between it and the former one is that the second fault will last for ω simulation steps and not always happens individually among

Parameter	Value	Parameter	Value
Temperature-proposal-selector layers	$[2+2n, 512, 256, 256, 128, 256, 256, 256, 2n]$	DQN layers	$[9+n, 50, 100, 200, 400, 16]$
Predictor-layers	$[(2+2n)k, 512, 256, 256, 256, 256, n]$	T_l	20 °C
T_{ofs}^{pre}	22	T_u	24.4 °C
l_{ft}	3	P_{pre}	0.1
β	6.25e-4	P_{sel}	0.3
η_0	1e-3	α	1e-3
η_2	1e-6	b_{MS}	40
L	5760	η_1	1e-4
Δt_s	1 min	η_3	1e-3
T_l^{out}	10 °C	k	20
v	2	Δt_c	15 min
l_{sel}	50	T_u^{out}	40 °C
m	2	$l_{r_{sel}}$	1e-4
γ	0.99	T_{air}	10 °C
l_{MS_i}	3	η_0	0.003
		b	32
		l_{MS}	2

Table 1: Hyper-parameters used in our experiments.

the time period. Thus this type of fault can cause larger damage to the system than the first one. And we call it continuous faults.

Metrics for evaluation: We evaluate the sensor fault-tolerant temperature control results based on the average indoor temperature violation rate θ_i for each thermal zone i and the total energy cost for running the HVAC system. We evaluate the performance of model-assisted learning on the temperature prediction task with a four-zone building. The measurement for the predictor is based on the normalized root mean square error (Normalized RMSE) between the prediction and the actual temperature value.

4.2 Experiment settings

The experiments are run on an Ubuntu OS server equipped with NVIDIA TITAN RTX GPU cards. The learning algorithm implementations are based on the Pytorch framework. The Adam optimizer [15] is utilized for all neural networks' training. We use the EnergyPlus [3] simulation tool to simulate the behavior of real buildings. Note that this is only for experimentation purpose. In practice, our tool will be deployed directly on real buildings with the modules trained on the data collected from those buildings. Moreover, the interaction between the building simulations in EnergyPlus and the Pytorch learning algorithms is implemented through the Building Controls Virtual Test Bed (BCVTB) [36]. We use a single-zone building and a 4-zone building as the target buildings for conducting our experiments, and the building simulation utilizes the summer weather data in August at Riverside, California, USA, which is obtained from the Typical Meteorological Year 3 database [37]. The comfort temperature bound is set to $[20^\circ\text{C}, 24.4^\circ\text{C}]$ based on the United States Occupational Safety and Health Administration (OSHA) recommendation [25]. The hyper-parameter settings mentioned in the previous sections are shown in Table 1.

4.3 Evaluation of sensor fault-tolerant framework on IID and continuous faults

This section shows the performance of our sensor fault-tolerant framework and its comparison with a standard DQN controller.

The experiments are conducted on a single-zone building and a four-zone building under different sensor fault patterns.

4.3.1 Against IID faults. We first study how much the sensor fault-tolerant framework can protect the control performance from the IID faults. The IID faults happen individually at each simulation step with the probability p_1 , and we test the case where p_1 is chosen from $[0, 0.1, 0.2, 0.4, 0.6, 0.8]$. The model is first tested on a single zone building. Table 2 shows the results comparison between the standard DQN controller (DQN) and our sensor fault-tolerant framework (FTF). We can see that the typical DQN controller's performance significantly deteriorates when facing the IID faults, as the heavily faulty sensor data nearly paralyzed the normal function of the neural network. The problem gets worse with the fault occurring probability p_1 becomes larger. For our sensor fault-tolerant framework, the average temperature violation rate remains very low under varying degree of IID faults (72.8% to 86.2% reduction in violation rate when compared with the standard DQN under fault probability from 0.1 to 0.8). Moreover, even with our approach's much more robust control, the energy cost does not increase much compared to the non-faulty case, which shows the cost-effectiveness of our sensor fault-tolerant approach.

We also tested our framework on a 4-zone building against the IID faults, and Table 3 shows its comparison with the standard DQN. θ_1 to θ_4 are the temperature violation rate for each of the 4 thermal zones. Again, we can clearly see that our approach can maintain the violation rate at a low level under varying level of sensor faults, and can significantly outperform the standard DQN (73.0% to 92.2% reduction in violation rate under fault probability from 0.1 to 0.8). It is also worth mentioning that when there is no fault, our framework will not introduce additional overhead. Finally, Fig. 3 also provides a visualization of the temperature change on the 4-zone building under IID faults with $p_1 = 0.4$ with/without the sensor fault-tolerant framework, and we can clearly see the effectiveness of our framework in keeping the temperature within comfort bound under faults.

4.3.2 Against continuous faults. We then evaluate our approach against continuous faults. Similar to what we have shown in the previous section, the model is tested on a single-zone building and a four-zone building, with the probability p_2 set to 0.2(single-zone), 0.1(four-zone) and ω selected from 0 to 5. The comparison between our approach and the standard DQN is presented in Table 4 and Table 5. The temperature violation rates in the tables are all higher than the previous section under the same fault probability, which indicates that the continuous faults can cause more damage than the IID faults. As shown in the table, the standard DQN controller drastically increases the violation rate for 4 $\times$ to 26 $\times$ for single zone and 9 $\times$ to 115 $\times$ for four-zone under continuous faults. In comparison, our approach can effectively maintain the violation rate at a low level (12.5% to 89.1% reduction for single zone and 73.0% to 86.6% reduction for four-zone in violation rate when compared with the standard DQN under fault time ω from 1 to 5).

4.4 Evaluation of model-assisted learning

In this section, we conduct experiments on the model-assisted learning algorithm and demonstrate its improvement in the performance

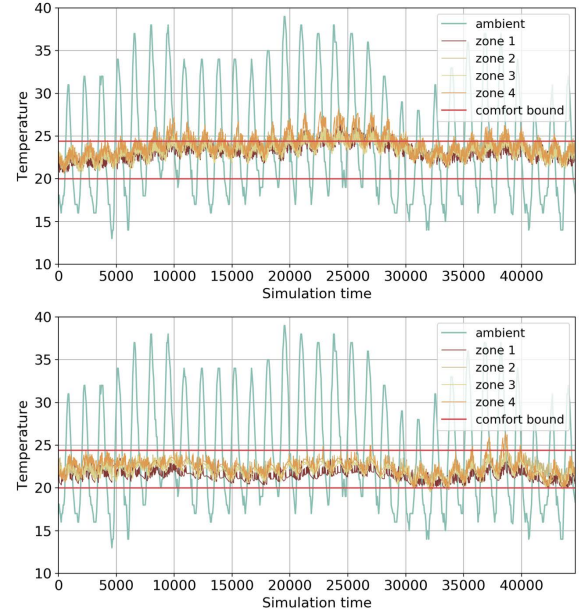


Figure 3: 4-zone building temperature under IID faults with $p_1 = 0.4$ without FTF (above) and with FTF control (below).

	p_1	0	0.1	0.2	0.4	0.6	0.8
DQN	θ	0.38	1.03	1.32	4.99	9.88	17.86
	Cost	235.19	241.20	237.64	228.50	226.40	223.81
FTF	θ	0.39	0.28	0.18	0.39	1.29	2.46
	Cost	247.60	247.30	248.17	250.74	254.81	265.52

Table 2: Comparison between standard DQN controller and our sensor fault-tolerant framework (FTF) on a single-zone building under IID faults. p_1 is the fault occurring probability. θ is the average indoor temperature violation rate (%).

	p_1	0	0.1	0.2	0.4	0.6	0.8
DQN	θ_1	0.0	0.04	3.32	9.47	22.03	21.66
	θ_2	0.12	1.32	5.79	20.58	35.50	40.79
	θ_3	0.11	0.40	4.46	11.55	19.94	24.66
	θ_4	0.43	4.27	15.62	34.90	47.84	50.42
	Cost	257.79	246.17	228.01	205.60	192.24	184.84
FTF	θ_1	0.0	0.0	0.0	0.35	7.12	8.91
	θ_2	0.17	0.31	0.17	0.0	0.96	1.87
	θ_3	0.03	0.34	0.15	1.18	5.65	6.35
	θ_4	0.39	0.98	0.74	2.46	5.67	6.58
	Cost	257.82	257.48	265.23	298.18	314.52	316.53

Table 3: Comparison between standard DQN controller and our sensor fault-tolerant framework (FTF) on a four-zone building under IID faults. p_1 is the fault probability. θ_i is the avg. indoor temperature violation rate (%) in thermal zone i .

of the temperature predictor module. Note that the data only contains non-faulty data in this section for avoiding other factors that may affect the evaluation, which means that there is no sensor fault in both training and testing.

We employ an abstract physical model introduced in Section 3.3 for a four-zone building. The abstract model itself has the temperature prediction value with Normalized RMSE at 3.7e-2. Then if we

	ω	0	1	2	3	4	5
DQN	θ	0.38	1.32	2.74	3.83	6.66	9.73
	Cost	235.19	237.64	232.48	232.16	229.21	225.79
FTF	θ	0.39	0.28	0.30	0.95	2.86	4.82
	Cost	247.60	247.30	248.14	245.93	244.10	242.88

Table 4: Comparison between standard DQN controller and our sensor fault-tolerant framework (FTF) on a single-zone building under continuous faults. The fault lasts for ω steps. θ is the avg. indoor temperature violation rate (%).

	ω	0	1	2	3	4	5
DQN	θ_1	0.0	0.04	3.02	5.27	8.07	10.96
	θ_2	0.12	1.32	4.71	11.51	15.32	20.58
	θ_3	0.11	0.40	3.57	5.89	10.00	12.63
	θ_4	0.43	4.27	13.24	24.48	26.98	31.52
	Cost	257.80	246.17	229.04	219.30	212.56	207.90
FTF	θ_1	0.0	0.0	0.0	0.0	0.0	0.23
	θ_2	0.17	0.31	0.88	1.77	2.94	5.32
	θ_3	0.03	0.34	0.17	0.82	0.82	1.95
	θ_4	0.39	0.98	1.18	2.92	4.31	7.54
	Cost	257.82	257.48	267.24	267.21	265.26	257.55

Table 5: Comparison between standard DQN controller and our sensor fault-tolerant framework (FTF) on a four-zone building under continuous faults. The fault lasts for ω steps. θ_i is the avg. indoor temperature violation rate (%) in thermal zone i .

Amount of data	360	720	1440	2880	5760
Labeled data only	2.86e-2	1.97e-2	1.17e-2	8.71e-3	6.02e-3
Distillation+Fine-tuning	2.85e-2	1.81e-2	1.13e-2	9.76e-3	6.55e-3
Model-assisted SSL	1.56e-2	1.03e-2	7.83e-3	5.01e-3	4.13e-3
Model-assisted RU	1.54e-2	1.19e-2	8.79e-3	6.42e-3	3.56e-3
Model-assisted learning	1.15e-2	9.94e-3	5.72e-3	2.29e-3	1.98e-3

Table 6: Comparison of different learning strategies on temperature predictor performance. The first line shows training with labeled data only. The second line shows the distillation approach as in [10]. The third line shows using only the first stage (model-assisted SSL) of our model-assisted learning approach, and the fourth line shows only using the second stage (model-assisted RU). The last line shows using both stages, i.e., our model-assisted learning approach.

only use the accurately labeled data collected from the building to train the network in the temperature predictor module, which is shown in the first line in Table 6 (*Labeled data only*), the Normalized RMSE remains at a relatively high level, e.g., $2.0\text{e-}2$ for 1440 data samples, and $1.8\text{e-}2$ for 2880 data samples. More labeled data leads to more accurate model prediction. The maximum amount of available data is 5760 samples for the simulation of eight days.

In addition to model-assisted learning, we also test another idea for leveraging the abstract physical model M to gain better performance, i.e., using the abstract physical model to set initial weights for a neural network, so the network may cost less training data for reaching higher accuracy as it searches from a better initial point. The related technique for obtaining this initial value is model distillation [10]. However, as mentioned earlier, choosing the data to feed the neural network is challenging for distillation. Here we use the same sampling approach as proposed in Section 3.3, i.e.,

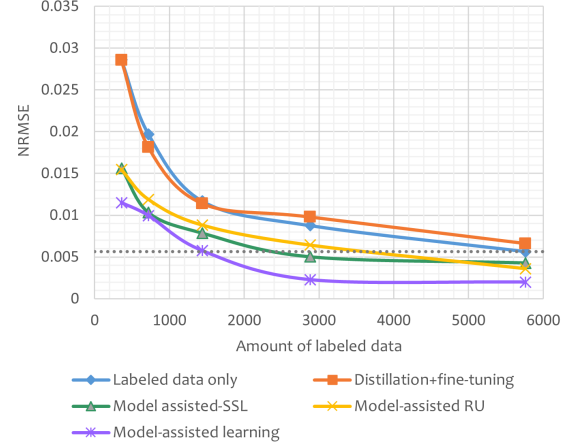


Figure 4: Comparison of different learning strategies on temperature prediction performance, including *Labeled data only* (blue line), *Distillation+Fine-tuning* (orange line), *Model-assisted SSL&RU* (green&yellow line), *Model-assisted learning* (purple line). We can observe from the figure that *Model-assisted learning* only requires around 1400 data samples to reach the performance of using *Labeled data only* with 5760 samples, i.e., only needs 1/4 of the labeled data by leveraging the abstract physical model via our approach.

sampling from data space H and feeding the samples x to the abstract physical model M . Then we get the corresponding data pair (x, y) , and train the network using (x, y) with learning rate η_2 for n_{iter} iterations (a new sampling data batch for each iteration). Next, we fine-tune this newly trained model with learning rate η_3 in l_{ft} epochs on the accurately labeled data. The model obtained in this way is named as *Distillation + Fine-tuning* (which is shown in the second line of the Table 6).

Finally, we apply our proposed model-assisted learning to leverage the abstract physical model. To understand how much each stage contributes to the final performance, we add the results of only applying one of the two stages, which are the third line (*Model-assisted SSL*) and fourth line (*Model-assisted RU*) of Table 6, respectively. And when combining both, the result is our *Model-assisted learning*, as in the last line.

We can observe from the table that, when the available sample is limited (360, 720, 1440), the building dynamics directly extracted by *Distillation + Fine-tuning* method can help reduce the Normalized RMSE. However, those extracted knowledge is only an inaccurate estimation, and the bias it brings prevents the model from achieving better result when there is more available labeled data (2880, 5760). On the other side, both stages in our *Model-assisted learning* approach can make good use of the abstract model and reduce the Normalized RMSE among all cases. When combining the two together, with the same amount of labeled data, our *Model-assisted learning* can achieve significantly better results than using only labeled data or distillation method. Such effectiveness is also visualized in Figure 4 – it plots the same results as Table 6, but we can clearly see that for the same level of performance, our *Model-assisted learning* approach only requires about 1/4 of the labeled data.

5 CONCLUSION AND FUTURE WORKS

In this paper, we present a novel learning-based sensor fault-tolerant control framework for building HVAC systems against faulty sensor readings, which includes neural network-based components for temperature prediction, temperature proposal selection, and DRL-based HVAC control. We also introduce a model-assisted learning approach that leverages the abstract physical model to overcome the difficulty in training data insufficiency. Experimental results demonstrate the effectiveness of our framework and the model-assisted learning method. In future, we plan to explore integrating other controllers into our framework to improve their fault-resilience under our predictor and selector design. We also plan to explore monitoring the zone occupancy and considering it in evaluating the discomfort cost of the reward function, for better performance and more energy saving.

ACKNOWLEDGMENTS

We gratefully acknowledge the support from Department of Energy (DOE) award DE-EE0009150 and National Science Foundation (NSF) awards 1834701, 1839511 and 2038853.

REFERENCES

- [1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.
- [2] 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.
- [3] Drury B Crawley, Linda K Lawrie, Curtis O Pedersen, and Frederick C Winkelmann. 2000. Energy plus: energy simulation program. *ASHRAE journal* 42, 4 (2000), 49–56.
- [4] Jonny Carlos da Silva, Abhinav Saxena, Edward Balaban, and Kai Goebel. 2012. A knowledge-based system approach for sensor fault modeling, detection and mitigation. *Expert Systems with Applications* 39, 12 (2012), 10977–10989.
- [5] Zhimin Du, Bo Fan, Jinlei Chi, and Xinqiao Jin. 2014. Sensor fault detection and its efficiency analysis in air handling unit using the combined neural networks. *Energy and Buildings* 72 (2014), 157–166.
- [6] Zhimin Du, Bo Fan, Xinqiao Jin, and Jinlei Chi. 2014. Fault detection and diagnosis for buildings and HVAC systems using combined neural networks and subtractive clustering analysis. *Building and Environment* 73 (2014), 1–11.
- [7] Jordi Fonollosa, Alexander Vergara, and Ramón Huerta. 2013. Algorithmic mitigation of sensor failure: Is sensor replacement really necessary? *Sensors and Actuators B: Chemical* 183 (2013), 211–221.
- [8] Guanyu Gao, Jie Li, and Yonggang Wen. 2020. DeepComfort: Energy-Efficient Thermal Comfort Control in Buildings via Reinforcement Learning. *IEEE Internet of Things Journal* 7, 9 (2020), 8472–8484.
- [9] Volkan Gunes, Steffen Peter, and Tony Givargis. 2015. Improving energy efficiency and thermal comfort of smart buildings with HVAC systems in the presence of sensor faults. In *2015 IEEE HPCSS-CSS-ICESS*. IEEE, 945–950.
- [10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
- [11] David G Holmberg and D Evans. 2003. *BACnet wide area network security threat assessment*. US Department of Commerce, National Institute of Standards and Technology.
- [12] Xinqiao Jin and Zhimin Du. 2006. Fault tolerant control of outdoor air and AHU supply air temperature in VAV air conditioning systems using PCA method. *Applied Thermal Engineering* (2006).
- [13] Woohyun Kim and Srinivas Katipamula. 2018. A review of fault detection and diagnostics methods for building systems. *Sci. Technol. Built Environ.* (2018).
- [14] Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947* (2016).
- [15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [16] 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.
- [17] Jie Li, Wei Zhang, Guanyu Gao, Yonggang Wen, Guangyu Jin, and Georgios Christopoulos. 2021. Towards Intelligent Multi-Zone Thermal Control with Multi-Agent Deep Reinforcement Learning. *IEEE IoT Journal* (2021).
- [18] Paulo Lissa, Michael Schukat, and Enda Barrett. 2020. Transfer Learning Applied to Reinforcement Learning-Based HVAC Control. *SN Computer Science* 1 (2020).
- [19] Jingjing Liu, Min Zhang, Hai Wang, Wei Zhao, and Yan Liu. 2019. Sensor fault detection and diagnosis method for AHU Using 1-D CNN and clustering analysis. *Computational intelligence and neuroscience* 2019 (2019).
- [20] Zhenjun Ma and Shengwei Wang. 2012. Fault-tolerant supervisory control of building condenser cooling water systems for energy efficiency. *HVAC&R Research* 18, 1-2 (2012), 126–146. <https://doi.org/10.1080/10789669.2011.568320>
- [21] 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.
- [22] Maryam Sadat Mirnaghi and Fariborz Haghighat. 2020. Fault detection and diagnosis of large-scale HVAC systems in buildings using data-driven methods: A comprehensive review. *Energy and Buildings* (2020), 110492.
- [23] Aviek Naug, Ibrahim Ahmed, and Gautam Biswas. 2019. Online energy management in commercial buildings using deep reinforcement learning. In *2019 IEEE SMARTCOMP*. IEEE, 249–257.
- [24] H Michael Newman. 2013. *BACnet: The Global Standard for Building Automation and Control Networks*. Momentum Press.
- [25] United States Department of Labor. 2021. OSHA Technical Manual (OTM) Section III: Chapter 2.
- [26] Panayiotis M Papadopoulos, Vasso Reppa, Marios M Polycarpou, and Christos G Panayiotou. 2018. Distributed Design of Sensor Fault-Tolerant Control for Preserving Comfortable Indoor Conditions in Buildings. *IFAC-PapersOnLine* 51, 24 (2018), 688–695.
- [27] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. 2015. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *ICCV*.
- [28] Jianying Qin and Shengwei Wang. 2005. A fault detection and diagnosis strategy of VAV air-conditioning systems for improved energy and control performances. *Energy and buildings* 37, 10 (2005), 1035–1048.
- [29] Vasso Reppa, Panayiotis Papadopoulos, Marios M Polycarpou, and Christos G Panayiotou. 2014. A distributed architecture for HVAC sensor fault detection and isolation. *IEEE Transactions on Control Systems Technology* (2014).
- [30] 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.
- [31] Yu-Yin Sun, Yin Zhang, and Zhi-Hua Zhou. 2010. Multi-label learning with weak label. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 24.
- [32] Mohamed Toub, Chethan R Reddy, Meysam Razmara, Mahdi Shahbakhti, Rush D Robinett III, and Ghassane Aniba. 2019. Model-based predictive control for optimal MicroCSP operation integrated with building HVAC systems. *Energy Conversion and Management* 199 (2019), 111924.
- [33] Shengwei Wang and Youming Chen. 2002. Fault-tolerant control for outdoor ventilation air flow rate in buildings based on neural network. *Building and Environment* 37, 7 (2002), 691–704.
- [34] Shengwei Wang and Jingtan Cui. 2005. Sensor-fault detection, diagnosis and estimation for centrifugal chiller systems using principal-component analysis method. *Applied Energy* 82, 3 (2005), 197–213.
- [35] Tianshu Wei, Yanzhi Wang, and Qi Zhu. 2017. Deep reinforcement learning for building HVAC control. In *Proceedings of the 54th annual DAC* 2017. 1–6.
- [36] 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.
- [37] Stephen Wilcox and William Marion. 2008. Users manual for TMY3 data sets. (2008).
- [38] Qingqing Xu and Stevan Dubljevic. 2017. Model predictive control of solar thermal system with borehole seasonal storage. *Computers & Chemical Engineering* 101 (2017), 59–72.
- [39] Shichao Xu, Yixuan Wang, Yanzhi Wang, Zheng O'Neill, and Qi Zhu. 2020. One for Many: Transfer Learning for Building HVAC Control. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 230–239.
- [40] Xue-Bin Yang, Xin-Qiao Jin, Zhi-Min Du, Bo Fan, and Yong-Hua Zhu. 2014. Optimum operating performance based online fault-tolerant control strategy for sensor faults in air conditioning systems. *Automation in Construction* 37 (2014).
- [41] 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.
- [42] Zhi-Hua Zhou. 2018. A brief introduction to weakly supervised learning. *National science review* 5, 1 (2018), 44–53.
- [43] Xiaojin Zhu and Andrew B Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning* 3, 1 (2009), 1–130.