NORTHWESTERN UNIVERSITY

Learning from Limited and Imperfect Data in Cyber-Physical System

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Shichao Xu

EVANSTON, ILLINOIS

June 2023

ABSTRACT

Machine learning is seeping into every fabric in various practical domains such as autonomous driving, wearable computing, and smart buildings. However, in the actual development and integration, especially when the learning-based components are frequently included as components of large complex systems where the physical instances can be included as interactable components, they often pose significant data challenges, including data noise, insufficient training data, or lacking annotations, which could significantly hinder the learning process. In this dissertation, we will introduce several approaches to tackle these challenges. In building-related cyber-physical systems, a transfer learning-based approach is proposed to tackle the challenge of long training time in model-free Deep Reinforcement Learning (DRL) for building HVAC control. Then we also consider accelerating online DRL for building HVAC control with the help of heterogeneous expert guidances. Besides, to handle the problem of HVAC control under corrupted sensor inputs, a learning-based framework is proposed for sensor fault-tolerant building HVAC control. In the vision task domain, we investigate the challenge of using the cross-domain unlabeled data and weak annotator to mitigate the data insufficiency in the target domain. Moreover, we also propose a novel approach for handling multi-label classification with unseen classes in the testing stage.

ACKNOWLEDGEMENTS

My Ph.D. journey has been a shining star in my life, and I will cherish every moment I spent at Northwestern University. From the excitement of publishing my first paper to the disappointment of receiving rejections, from the laughter-filled conversations to the moments of stress preparing for the exams, and from the joy of hanging out around Evanston to the mixed emotions of receiving my Ph.D. degree, I have had wonderful experiences that will stay with me forever.

I will miss the time when we gathered together and had BBQ near the beach, the food in the Northwestern University cafeterias, the brainstorming sessions for new paper ideas, the talks, the chats, the coffee breaks that made each day so much enjoyable. Although it's a pity that the pandemic prevented me from traveling around the world to attend various academic conferences, I am grateful for the opportunities to complete internships at OPPO and Google, which broadened my horizon and expanded future career opportunities. What I miss the most is the fantastic and amazing people I met during my Ph.D. career. The memories, wishes, and connections I made during the time at Northwestern University will stay with me for my next life phase.

I would like to express my deepest appreciation to my advisor and dissertation committee chair, Prof. Qi Zhu, for his unwavering support and guidance throughout my Ph.D. journey. I can still remember the time when I was stressed and hurry to publish a paper in my first year, Prof. Zhu told me to slow down and explore the topics I am interested. In the later years, he taught me how to do academic research, how to identify the future opportunities for researching, how to write paper, how to present the idea, how to give a talk, etc. I was gradually on the right track, and qualified for being a true Ph.D.. Prof. Zhu was always patient and willing to help me both academically and professionally.

I am also grateful to my current and previous collaborateors, including Prof. Zheng O'Neill and Dr. Yangyang Fu from Texas A&M University, Prof. Xiao Wang, Prof. Josiah Hester and Prof. Zhaoran Wang from Northwestern University, Dr. Zhuoran Yang from Yale University, Prof. Chao Huang from University of Liverpool, Prof. Yanzhi Wang from Northeastern University, etc. Their valuable suggestions, ideas and expertise in their respective research areas were crucial in helping me work on various projects.

Then I extend my thanks to the professors and tutors in the ELP department. They helped adjust to life in Evanston during my first month in the United States and also provided services to improve my English speaking skills during my first two years.

Moreover, I would like to express my gratitude to my colleagues at Northwestern University who I have a good time with during my Ph.D.. They are Zhilu Wang, Shuyue Lan, Hengyi Liang, Xiangguo Liu, Yixuan Wang, Ruochen Jiao, Lixu Wang, Payal Mohapatra, and Anthony J Goeckner. It's my great fortune to meet with these lovely people, collaborate with them on various research works, share the ideas, and enjoy the campus life.

Besides, I am honored to have Prof. Qi Zhu, Prof. Xiao Wang, and Prof Jie Gu as my dissertation committee members. Thank you for generously dedicating your time to my dissertation committee.

Last but most importantly, I express my deepest gratitude to my family for their unwavering support and understanding throughout my Ph.D. journey. Their love and care were instrumental in fortifying my conviction to begin and complete my Ph.D. studies.

We gratefully acknowledge the support from Richter Memorial Fund for terminal year fellowship, and also thanks Department of Energy (DOE) award DE-EE0009150 and National Science Foundation (NSF) grants 1834701, 1839511, 1724341, 2038853, 2016240, Office of Naval Research grant N00014-19-1-2496, and research awards from Facebook, Google, PlatON Network, and General Motors.

Abbreviation	**Definition**
ARX	AutoRegressive models with eXogenous inputs
A3C	Asynchronous Advantage Actor-Critic
BCQ	Batch-Constrained Q-Learning
CQL	Conservative Q-Learning
DA	Domain Adaptation
DDPG	Deep Deterministic Policy Gradient
DRL	Deep Reinforcement Learning
DQN	Deep Q-Network
DDQN	Double Deep Q-Network
FCU	Fan Coil Unit
HVAC	Heating, Ventilation, and Air Conditioning
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
ML	Machine Learning
MDP	Markov decision process
NLP	Natural Language Processing
PPO	Proximal Policy Optimization
PuL	Positive-unlabeled Learning
SOTA	State-Of-The-Art
SSL	Semi-Supervised Learning
UDA	Unsupervised Domain Adaptation
VLP	Vision-Language Pretraining
VAV	Variable Air Volume

ABBREVIATIONS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Background and Motivation

Deep neural network (DNN), inspired by the connection of the neurons in the human nervous system, is by far the most effective model among all machine learning algorithms. It demonstrates strong representation learning ability and can handle a wide range of data modalities. Its applications are diverse and not limited to digital applications like video games [1], image rendering [2], music composition [3], but also extend to cyber-physical systems (CPS) where the physical components are involved, such as autonomous driving [4], high-energy physics [5], smart building [6], agriculture [7], etc. People's daily life has been greatly changed by the benefits brought from these advancements, and an increasing number of researches are also made in this field which reflects its importance and the potential benefits.

Image classification is a crucial component in many CPS systems, and its accuracy has improved significantly over the years, thanks to advancements in learning algorithms, computation hardware, and the increasing amount of training data. When we look at the typical human-designed models with supervised learning, the previous model AlexNet [8] and VGG [9] demonstrate the strong recognition ability of the DNN when trained on millions of accurately labeled images. Subsequent models, such as ResNet [10], DenseNet [11], ViT [12], and Swin Transformer [13], have outperformed human recognition ability in some specific domains. These models use convolutional layers or transformer blocks and heavily rely on massive amounts of training data. In addition, there have been explorations based on unsupervised (self-supervised) learning methods, including SimCLR [14] , MoCo [15], MAE [16], as well as studies that leverage information from different modalities, such as CLIP [17], ALIGN [18], GPT-4 [19], etc. The success of these large models heavily replies on the massive amount of training data.

Another field that has benefited greatly from neural networks is deep reinforcement learning (DRL), which aims to learn an optimal policy from interactions with the target environment. There are a number of successful cases in the literature which leverage DNN as the value function approximator or system state encoder, such as deep Q-learning (DQN) [1], quantile regression-DQN (QRDQN) [20], asynchronous advantage actor-critic algorithms (A3C) [21], Deep Deterministic Policy Gradient(DDPG) [22], Proximal Policy Optimization(PPO) [23], etc. After repeatedly trials on the target environment, these methods are able to learn a good control policy from past experiences.

However, compared to humans who possess remarkable abilities to extract knowledge and learn from limited or noisy observations to handle complex tasks, current machine learning models still face significant challenges when face the same scenarios. When considering machine learning models as components of large complex systems, it is important to take into account various factors as real-world scenarios can deviate from ideal conditions. Some models may struggle to achieve desired performance when faced with disturbances or malicious attacks, which are common in real-world CPS systems. Other models may require a large amount of training data, either labeled or unlabeled, which can be challenging to obtain due to the expensive data collection process in most of CPS systems.

In the field of building-related researches, there is a growing concern about developing HVAC controllers with minimal effort. Since the model-based methods require sufficiently accurate physical models for optimal performance, researchers make further exploration on model-free DRL to design the controller. For instance, the Wei et al. [6] leverage DQN methods to design the building

HVAC controller based on training with EnergyPlus building simulation platform. In [24], [25], the actor-critic methods in model-free DRL are applied. However, running the experiments in real building is time-costly, expensive and risky, it can also be inefficient to train on the building simulation platform [26] due to its complexity. Thus, when applying model-free DRL methods to building HVAC systems, the challenge of data insufficiency arises, leading to long training time. For example, in [27], it took DDPG around 2.4×10^4 months to achieve the best performance for temperature control and energy management. In [28], for a multi-zone building environment, the training time was extended to 4×10^4 months. Furthermore, the typical building HVAC systems may also suffer from imperfect data challenge caused by corrupted sensor readings, which can lead to performance degradation of the HVAC controller.Previous works have attempted to detect and correct these faults using model-based methods. For instance, Papadopoulos et. al. [29] develop their solution based on previously built complex physical building model as well as a fault model based on the assumption that sensor faults occur in a single zone at each time. However, in real building scenarios, modeling both the faults and the building dynamics can be difficult, making fault prediction or correction even more challenging.

Besides, vision plays a crucial role in many CPS applications, and we also investigate the data limitation challenge in image classification tasks. Numerous image classification researches (such as ResNet [10], DenseNet [11], ViT [12], etc.) have been carried out with the assumption that accurately labeled target domain data is available. However, gathering ample data samples can be challenging in some problem domains or scenarios, such as for the training of autonomous vehicles during extreme weather (e.g., fog, snow, hail) and natural disasters (e.g., mudflow), or for search and rescue robots during forest fire and earthquake. In some extreme case, the target classes may not even appear in the training dataset. Thus, how to leveraging other information source to help the target domain learning is critically important.

1.2 Dissertation Contributions

In this dissertation, we explore two domains in which data limitations pose significant challenges. In building task domain, we firstly address the challenge of long training time in model-free DRL for building HVAC control, and then consider the HVAC control under corrupted sensor inputs. In vision task domain, we investigate the challenge of using the cross-domain unlabeled data and weak annotator to mitigate the data insufficiency in target domain, and also study the multi-label classification under the zero-shot setting.

Throughout my Ph.D. career, I have made substantial contributions to addressing the challenges of learning from limited and imperfect data in CPS systems. Specifically, I have focused on the limitations of data availability and data quality, and have developed novel approaches that leverage additional information sources to achieve reasonable performance for target tasks.

1.2.1 Data Availability

Chapter 2.1 addresses the challenge of slow and expensive data collection process in the real or simulated building HVAC control application, which has hindered the use of model-free deep reinforcement learning due to the long training time required on a physical system. Thus, we present a transfer learning-based approach [30] to overcome this challenge, and the algorithm 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 the 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.

While transfer learning is quite effective when the source and target buildings share a certain degree of similarity, it may not be feasible when there is no source building available or the target building is significantly different. Thus, another efficient learning strategy is still required for the control agent. In Chapter 2.2, we propose a unified learning framework [31] that leverages the knowledge from domain experts in various forms to accelerate online reinforcement learning for building HVAC control. It leverages the abstract physical models (e.g., RC-networks [32], ARX models [33], surrogate model based on DNN) of building thermal dynamics (they are not accurate enough for enabling training DRL or designing model-based methods with good performance, but nevertheless contain valuable information of building dynamics), historical data collected from existing controllers (they may not be able to train DRL controllers with good performance due to distribution shift, but also contain useful information on building behavior), and expert rules that reflect basic policies.

Not only for the building HVAC control applications, we also explore the data availability limitation in vision-related tasks. In Chapter **3-2**, we present an advanced technique in open-vocabulary multi-label image classification (Open-vocabulary is a generalization of zero-shot and weakly supervised settings and is more suitable for dealing with unseen classes). Our method [34] is built based on the novel Dual-Modal decoder (DM-decoder) with alignment between visual and textual features. The experiment results indicate that our method significantly outperforms the previous methods and reach the state-of-the-art level.

1.2.2 Data Quality

In the context of data quality, we examine a scenario in building HVAC systems that could suffer from various sensor faults and be susceptible to malicious attacks. Such faulty sensor inputs may lead to the violation of indoor environment requirements (e.g., temperature, humidity, etc.) and an increase in energy consumption. So we present a learning-based framework for sensor fault-tolerant HVAC control [35] which includes three deep learning-based components that can effectively compensate for sensor faults, ensuring that the building's temperature and other environmental requirements are met while maintaining energy efficiency. Our experiments demonstrate that the proposed framework can significantly reduce building temperature violations under a variety of sensor fault patterns, making it a promising solution for improving the reliability and robustness of building HVAC control systems.

In addition, we also examine a novel scenario in the vision task, where both data availability and quality are crucial factors to consider. In such cases, data from related domains and weak annotators can be employed to address data insufficiency in the target domain. To be specific, we propose weak adaptation learning (WAL) [36] approach that leverages unlabeled data from a similar source domain, a low-cost weak annotator (which produces labels based on task-specific heuristics, labeling rules, or other methods (albeit with inaccuracy)), and a small amount of labeled data in the target domain to learn an accurate classifier in the target domain.

The rest of the prospectus is organized as follows. Chapter 2 contains the studies related to building HVAC control. Chapter 2.1 introduces the methodology of transfer learning for the building HVAC system. Chapter 2.2 shows the learning acceleration framework for DRL method which leverages the knowledge from domain experts in various forms. Chapter 2.3 demonstrate a learning-based sensor fault-tolerant control framework for the building HVAC control. Then Chapter 3 includes the studies in vision tasks. Chapter 3.2 presents the weak adaptation learning method for addressing the cross domain data insufficiency. In Chapter 3.1, the new approach for open-vocabulary multi-label classification. We list the discussion and future directions in Chapter 4.

CHAPTER 2
ADDRESSING DATA CHALLENGES IN BUILDING HVAC CONTROL

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]. In addition, the operation of HVAC systems significantly affects the physical and mental health of building occupants, as people spend around 87% of their time indoors [30], [37], and even higher during the COVID-19 pandemic in recent years [38]. Thus, it is critically important to design effective HVAC control strategies that are both energy efficient and able to maintain the desired temperature and indoor air quality for occupants.

However, data challenges always exist during building HVAC control, the typical challenges include insufficient training data (or long training/data collection time), corrupted sensing data, etc. Thus, in this chapter, we will introduce three topics on the data challenges in building HVAC control,

- **Transfer learning for building HVAC control:** We propose a transfer learning approach that decomposes the design of a 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. The experiments 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. It 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.

- **Sample-efficient DRL for building HVAC control with heterogeneous expert guidances:** We propose a novel training framework to accelerate online RL for building HVAC control with heterogeneous expert guidances, including abstract physical models, historical data, and expert rules. These various guidances are unified in our framework via the expert functions. The experiment results demonstrate that our approach can effectively reduce the DRL training time while maintaining low energy cost and temperature violation rate. We also propose a novel runtime shielding framework with an expert model that can further reduce the temperature violation rate when applied to our learned DRL-based controller.
- **Sensor fault-tolerant building HVAC control:** We present a novel sensor fault-tolerant learning-based framework to achieve sensor fault resilience in 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. Then 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.

2.1 One for many: Transfer learning for building HVAC control

2.1.1 Background

In the literature, there is an extensive body of work addressing the control design of building HVAC systems [39]–[42]. 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 [40], [43], [44], and

Section 2.1 is based on our work published at [30].

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 [45].

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 [46]–[48], earlier works that utilize neural networks [49], [50], and more recent deep reinforcement learning (DRL) methods [6], [24], [25], [51]–[54]. 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 [45]. 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 [6], [45] and 4000+ months of data are used for more complex models [28], [54] – 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 [6], [45] thus propose to first use detailed and accurate physical models (e.g., EnergyPlus [55]) 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 chapter, 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.

2.1.2 Related Works

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 [39], [40]. For instance, the work in [56] 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 [40] 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 [41], [43], [44] 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 [55], 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 [6], [51], DRL is applied to building HVAC control and an EnergyPlus model is leveraged for simulation-based offline training of DRL. In [24], [25], DRL approaches leveraging the actor-critic methods are applied. The works in [53], [54] 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 [6], [45] or even longer [28], [54]. 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 us 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 [57], 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 [58], 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.

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 [59]–[62]. For instance, in [60], the distribution of optimal trajectories across similar robots is matched for transfer learning in robotics. In [62], 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.

2.1.3 Methodology

We present our transfer learning approach in this section, including the design of the two-subnetwork controller and the training process. Section 2.1.3.1 introduces the system model. Section 2.1.3.2 provides an overview of our methodology. Section 2.1.3.2 presents the design of the building-agnostic front-end (sub-)network, and Section 2.1.3.2 explains the design of the building-specific back-end (sub-)network.

2.1.3 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 [6], an n-zone building model with a variable

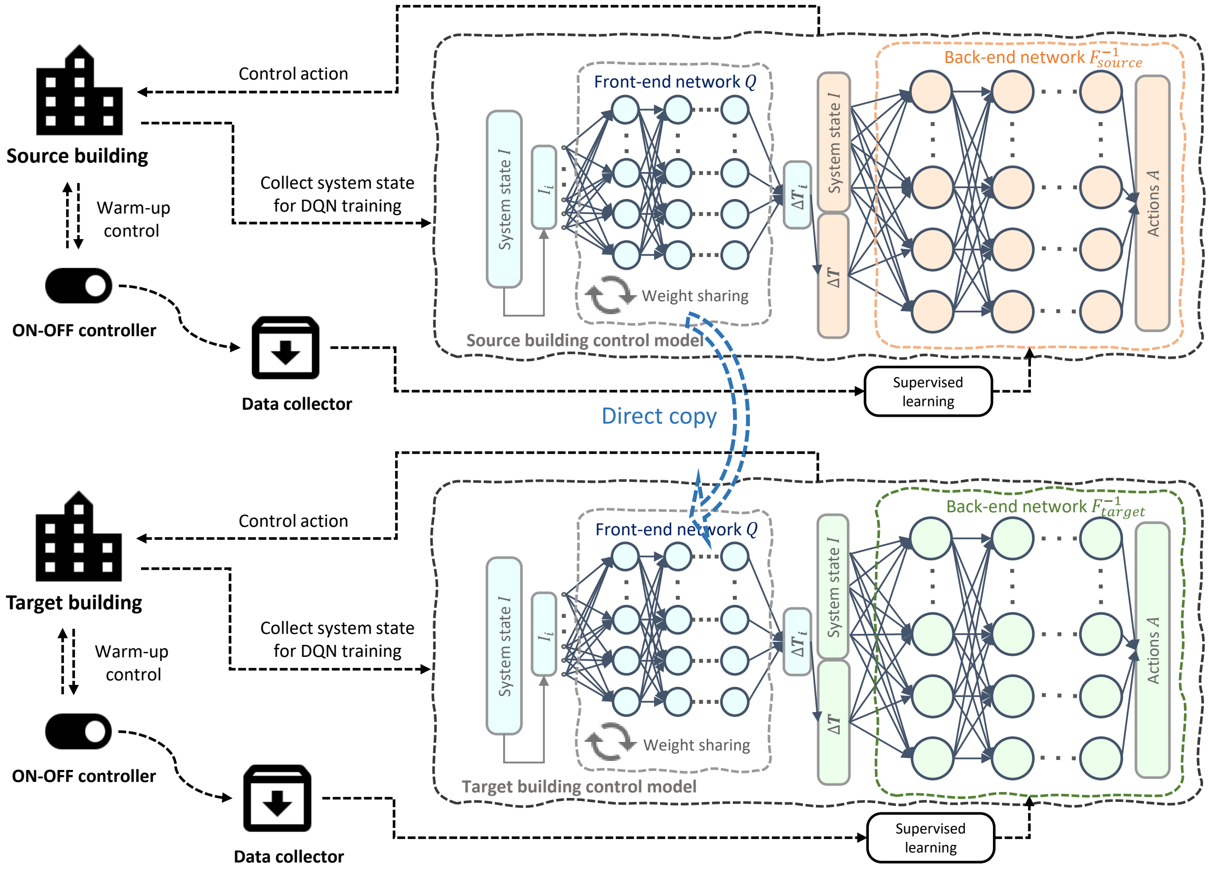


Figure 2.1: Overview of our DRL-based transfer learning approach for HVAC control. We design a novel DQN architecture that includes two sub-networks: A **front-end network** Q captures the building-agnostic part of the control as much as possible, while a **back-end network** (inverse building network) 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 $\mathbf{A}$. 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}.

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 [6], to improve DRL performance, S_{out} not only includes the current values of the ambient temperature T^i_{out} and the solar irradiance Sun^i_{out}, but also their weather forecast values for the next three days. Thus, the outside environment state is denoted as $S_{out} = \{T^0_{out}, T^1_{out}, T^2_{out}, T^3_{out}, Sun^0_{out}, Sun^1_{out}, Sun^2_{out}, Sun^3_{out}\}$. Our current model does not consider internal heat generation from occupants, a limitation that we plan to address in future work.

2.1.3 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 2.2 of Section 2.3.4), 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 2.1.

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 2.1 of Section 2.3.4). 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 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:

$$O_{\Delta T_i} = \begin{cases} < 1, 0, \cdots, 0 >, & \Delta T_i \leq -b, \\ < 0, \cdots, 0, 1, 0, \cdots, 0 >, & -b < \Delta T_i < b, \\ \scriptstyle the\ position\ of\ 1\ is\ at\ (\lfloor \Delta T_i/(2b/h) \rfloor) & \\ < 0, \cdots, 0, 1 >, & \Delta T_i \geq b. \end{cases} \tag{2.1}$$

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 [63], 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 [63]. 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 2.1.3.2. Specifically, the training of the front-end network Q follows Algorithm 1 (the hyper-parameters used are listed later in Table **??** of Section 2.3.4).

First, we initialize Q by following the weights initialization method described in [64] 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 2.1.3.2). 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}, ΔT, $\mathbf{A}$, R) and store them in the replay buffer. In the following learning-related operations, we first sample a data batch $M = (\mathscr{S}_{prime}, \mathscr{S}_{next}, ,)$ from the replay buffer, and calculate the actual temperature change value $\Delta\mathscr{T}_a$ from $\mathscr{S}_{prime}$ and $\mathscr{S}_{next}$. Then, we get the planned temperature change value from the back-end network F^{-1}, i.e., $_p$ = $F^{-1}(\Delta\mathscr{T}_a, \mathscr{S}_{prime})$. In this way, the cross entropy loss can be calculated from the true label and the predicted label $_p$. We then use supervised learning to update the back-end network F^{-1} with the Adam optimizer [65] under learning rate lr_2.

We follow the same procedure as described in [63] to calculate the target vector v that is used in deep Q-learning. With target vector v and input state S_{prime}, we can then train Q using the back-propagation method [66] 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 $\epsilon-$greedy policy to select each planned temperature change value ΔT_i:

$$\Delta T_i = \{ \begin{array}{ll} argmax\ O_{\Delta T_i} & with\,probability\ 1-\epsilon, \\ random(0\ to\ h+1) & with\,probability\ \epsilon. \end{array} \tag{2.2}$$

$$\Delta T = \{\Delta T_1, \Delta T_2, \cdots, \Delta T_n\}. \tag{2.3}$$

Algorithm 1 Training of front-end network Q

```
1: $ep$: the number of training epochs
2: $\Delta ct$: the control period
3: $t_{MAX}$: the maximum training time of an epoch
4: $\Delta nt$: 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 $\Delta T$
8: Initialize previous state $S_{pre}$
9: Initialize exploration rate $\epsilon$
       $Epoch$ = 1 to $ep$ $t$ = 0 to $t_{MAX}$, $t$ += $\Delta ct$
10: $S_{cur} \leftarrow$ ($t$, $S_{in}$, $S_{out}$)
11: Calculate reward $R$
12: Add experience ($S_{pre}$, $S_{cur}$, $\Delta T$, $\mathbf{A}$, $R$) to the replay buffer $tr$ = 0 to $L_{MAX}$
13: Sample a batch $M = (\mathscr{S}_{prime}, \mathscr{S}_{next}, , )$
14: Calculate actual temperature change value $\Delta \mathscr{T}_a$
15: Predicted label $_p$ = $F^{-1}(\Delta \mathscr{T}_a, \mathscr{S}_{prime})$
16: Set loss $L = CrossEntropyLoss(_p, )$
17: Update $F^{-1}$ with loss $L$ and learning rate $lr_2$
18: Target $\leftarrow$ target network $Q^{'}(\mathscr{S}_{prime})$
19: Train network $Q$ with $\mathscr{S}_{prime}$ and $t$ mod $\Delta nt$ == 0
20: Update target network $Q^{'}$
21: $O_{\Delta T} = Q(S_{cur})$
22: Update exploration rate $\epsilon$
23: Update each $\Delta T_i$ follows $\epsilon-$greedy policy
24: $\Delta T =< \Delta T_1, \Delta T_2, \cdots, \Delta T_n >$
25: Control action $\mathbf{A} \leftarrow F^{-1}(\Delta T, S_{cur})$
26: $S_{pre} = S_{cur}$
```

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

$$\mathbf{A} = F^{-1}(\Delta T, S_{cur}). \tag{2.4}$$

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 2.1 of Section 2.3.4). 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:

$$R_t = w_{cost} R\_cost_t + w_{vio} R\_vio_t, \tag{2.5}$$

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:

$$R\_cost_t = -\ cost(F^{-1}(\Delta T_{t-1}), t-1). \tag{2.6}$$

$$R\_vio_t = -\sum_{i=1}^{n} max(T_t^i - T_{upper}, 0) + max(T_{lower} - T_t^i, 0). \tag{2.7}$$

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 2.1). While such 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 [64]. 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 $\Delta\mathscr{T}$ 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 described in Algorithm 3. Note that we could further fine-tune our DRL controller during the

Algorithm 2 Training of back-end network F^{-1}

```
1:  $ep_F$: the number of training epochs
2:  $\Delta ct$: 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 $D$
        $t$ = 0 to $t_{MAX}$, $t$ += $\Delta ct$
7:  $S_{cur} \leftarrow (t, S_{in}, S_{our})$
8:  Control action $\mathbf{A} \leftarrow$ run ON-OFF controller on $S_{cur}$
9:  $S_{pre} = S_{cur}$
10: Add sample $(S_{cur}, S_{pre}, \mathbf{A})$ to database $M$
        each sample $\mathbf{u}$=$(S_{cur}, S_{pre}, \mathbf{a})$ in $M$
11: $\Delta\mathscr{T}_a \leftarrow$ calculate temperature difference in $(\mathscr{S}_{cur}, \mathscr{S}_{pre})$
12: Add sample $\mathbf{v} = (\Delta\mathscr{T}_a, S_{pre}, \mathbf{a})$ to dataset $D$
        each sample $\mathbf{u}$=$(S_{cur}, S_{pre}, \mathbf{a})$ in $M$
13: $\Delta\mathscr{T}_a \leftarrow$ lowest level
14: $\mathbf{a}^{'} \leftarrow$ maximum air condition level
15: Add sample $\mathbf{v} = (\Delta\mathscr{T}_a, S_{pre}, \mathbf{a}^{'})$ to dataset $D$ $Epoch$ = 1 to $ep_F$ each training batch of $(\Delta\mathscr{T}_a, S_{pre}, \mathbf{a})$
    in dataset $D$
16: network inputs = $(\Delta\mathscr{T}_a, S_{pre})$
17: corresponding labels = $(\mathbf{a})$
18: Train network $F^{-1}$
19: Return $F^{-1}$
```

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 [64], 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.

Algorithm 3 Running of our proposed approach

```
1: $\Delta ct$: 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 2.1)
4: Initialize the weights of $F^{-1}$ with weights learned using Algorithm 2
       $t$ = 0 to $t_{MAX}$, $t$ += $\Delta ct$
5: $S_{cur} \leftarrow (t, S_{in}, S_{out})$
6: $\Delta T \leftarrow argmax\ Q(S_{cur})$
7: Control action $\mathbf{A} \leftarrow F^{-1}(\Delta T, S_{cur})$
```

2.1.4 Experimental Results

2.1.4 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 [65] is used to optimize both front-end networks and back-end networks. The DRL hyper-parameter settings are shown in Table 2.1. In addition, to accurately evaluate our approach, we leverage the building simulation tool EnergyPlus [55]. 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 [6], [45], 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) [67]. We simulate the building models with the weather data obtained from the Typical Meteorological Year 3 database [68], and choose the summer weather data in August (each training epoch contains

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 2.1: Hyper-parameter setting of transfer learning for building HVAC control.

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 2.2. In addition, the conditioned air temperature sent from the VAV HVAC system is set to 10 °C.

The symbols used in the result tables are explained as follows. θ_i denotes the temperature violation rate in the thermal zone i. Aθ and Mθ 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 °C. 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 $\checkmark$; otherwise, it is $\times$ (which is typically not acceptable for HVAC control).

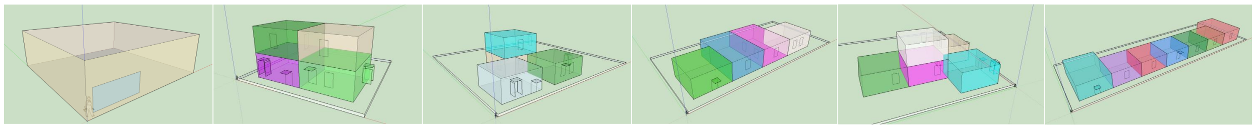


Figure 2.2: Different building models used in our experiments. From left to right, the models are one-zone building 1, four-zone building 1, four-zone building 2 , four-zone building 3, five-zone building 1, seven-zone building 1. Compared to four-zone building 1, four-zone building 2 has different layout 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	Target	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 2.2: 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.

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 2.1.3.2. Here as shown in Table 2.2, 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.

2.1.4 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 2.2, 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 2.3 first shows the direct training results on four-zone building 1, and the main transferring results are presented in Table 2.4.

The direct training outcome by baselines and our approach are shown in Table 2.3. The results include ON-OFF control, Deep Q-network (DQN) control as described in [6] (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 [69]), 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 2.4 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 2.3) to the target building. From Table 2.4, 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 2.1.4.5.

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[6]	*4-zone 1*	50	1.21%	22.72%	9.47%	20.66%	0.68	2.46	1.61	2.07	×	245.08
DQN[6]	*4-zone 1*	100	0.0%	0.53%	0.05%	0.93%	0.0	0.46	0.40	1.09	✓	292.91
DQN[6]	*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 2.3: 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[6]	*4-zone 2*	50	0.83%	49.22%	46.75%	60.48%	0.74	2.93	3.18	3.39	×	258.85
DQN[6]	*4-zone 2*	100	0.0%	1.67%	1.23%	3.58%	0.0	0.92	0.77	1.62	✓	352.13
DQN[6]	*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[6]	*4-zone 3*	50	0.68%	47.21%	44.61%	56.19%	0.74	3.15	2.92	3.60	×	267.29
DQN[6]	*4-zone 3*	100	0.34%	2.53%	2.21%	5.59%	0.01	1.18	0.85	1.18	×	342.08
DQN[6]	*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 2.4: 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 [6] 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.

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[6]	*5-zone 1*	50	38.65%	65.00%	2.60	3.81	×	263.79
DQN[6]	*5-zone 1*	100	4.13%	11.59%	4.66	1.47	×	326.50
DQN[6]	*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[6]	*7-zone 1*	50	28.14%	54.28%	2.76	3.06	×	248.38
DQN[6]	*7-zone 1*	100	5.19%	18.91%	1.12	1.69	×	277.87
DQN[6]	*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 2.5: Comparison of our approach and baselines on five-zone building 1 and seven-zone building 1.

2.1.4 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 2.5. 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 [6] also leads to high violation rate. In comparison, our approach achieves both low violation rate and low energy cost.

2.1.4 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

Method	AC	EP	$A\theta$	Mθ	Aμ	Mμ	☑	Cost
ON-OFF	AC 2	**0**	**0.15%**	**0.23%**	0.05	0.08	✓	329.56
DQN[6]	AC 2	50	20.28%	35.56%	1.73	2.66	×	229.41
DQN[6]	AC 2	100	1.25%	2.69%	0.61	1.20	✓	270.93
DQN[6]	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[6]	AC 3	50	2.85%	3.76%	1.37	1.90	✓	321.03
DQN[6]	AC 3	100	0.69%	1.20%	0.53	0.99	✓	**265.46**
DQN[6]	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 2.6: Results comparison under different HVAC equipment.

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 2.6. 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.

2.1.4 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 2.3.

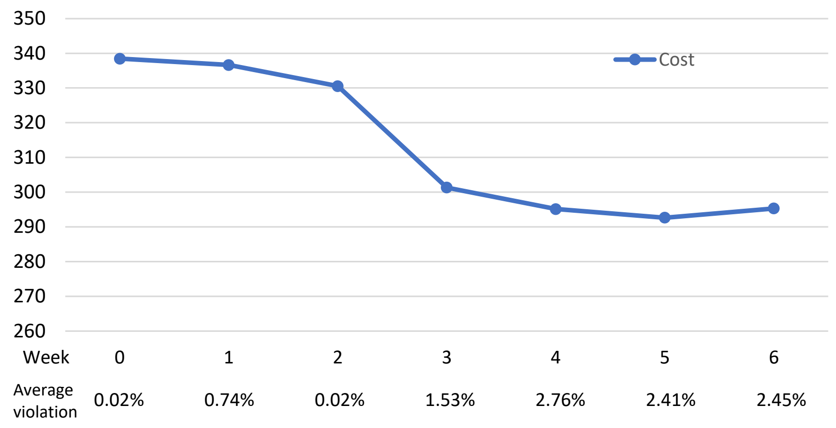


Figure 2.3: Fine-tuning results of our approach for four-zone building 2. Our approach can significantly reduce energy cost after fine-tuning for 3 weeks, while keeping the temperature violation rate at a low level.

2.1.4 Discussion

Transfer from n-zone to n-zone with Different Weather

As presented in [57], 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. We tested the weather from Riverside, Buffalo, and Los Angeles, which is shown in Figure 2.4. 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.

Different Settings for ON-OFF Control

Our back-end network (inverse building network) is learned from the dataset collected by an ON-

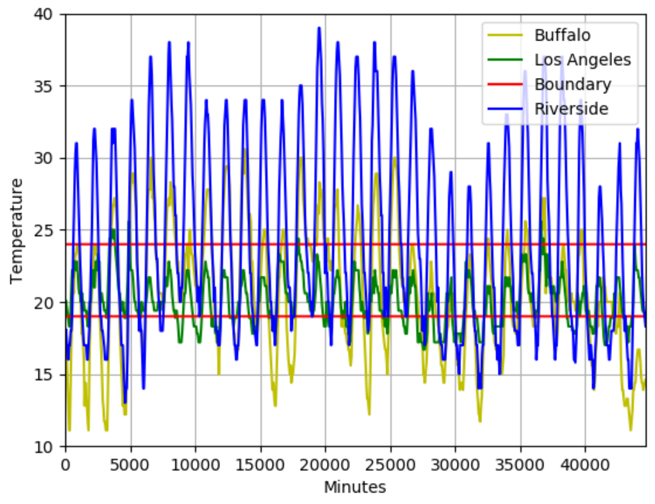


Figure 2.4: The visualization of different weathers. The yellow line is the Buffalo weather, the green line is the LA weather, the blue line is the Riverside weather, and the red lines are the comfortable temperature boundary.

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 2.7: 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 2.8: Results of testing using different boundary.

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 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 2.8 shows that with different boundary settings, supervised learning can stably learn from building-specific behaviors.

2.2 Accelerate Online Reinforcement Learning for Building HVAC Control with Heterogeneous Expert Guidances

2.2.1 Background

From the previous section, we know that a major difficulty in adopting DRL-based methods for building HVAC control is that it could take a long time to train the RL agent in practice during building operation. For instance, it may take more than 100 months of training to reach convergence for the Q-learning based methods [6], [70], and around 500 months of training for the DDPG algorithm to converge on a laboratory building model [54]. In [27], DDPG is used for temperature control and energy management, and it takes around 2.4×10^4 months to reach the best perfor-

Section 2.2 is based on our work published at [31].

mance. In [28], the training time is almost 4×10^4 months in a multi-zone building environment. Clearly, such long training time would make it impossible to adopt DRL in practice for building control. While developing a detailed simulation model (e.g., in EnergyPlus) and conducting the training via simulation may help avoid this issue, the development of the simulation model itself is difficult and costly (in terms of both time and expertise), just as in the model-based methods.

Thus, researchers have been trying to improve the training efficiency for DRL-based building HVAC control. In previous section [30], a transfer learning approach is proposed to extract and transfer the building-agnostic knowledge from an existing DRL controller of a source building to a new DRL controller of a target building, and only re-train the building-specific components for the new DRL controller. The work in [71] also leverages transfer learning, but for heat pump control in microgrid. However, the effectiveness of the transfer learning-based methods strongly relies on the similarity between the target building and the transferred building, and may not be feasible when they do not share many similarities. There are also a few studies on the application of offline reinforcement learning for building HVAC control, where historical data on existing controllers are leveraged to train new RL-based controllers. For instance, the work in [72] conducts conservative Q-learning (CQL) to train controllers for maintaining the room temperature setpoint. The problem of such offline RL methods, however, is that the learned agents' performance strong depends on the quality of the historical data. And they tend to perform poorly due to the distributional shift between the historical data and the learned policy, and may have limited improvement even with fine tuning via online training [73].

In this section, to address the above challenge in DRL training efficiency, we propose a **unified framework that leverages the knowledge from domain experts in various forms** to accelerate online RL for building HVAC control. This is motivated by the observation that in established domains such as building control, there is extensive domain expertise, represented in various forms such as 1) *abstract physical models* (e.g., RC-networks [32] or ARX models [33]) of building thermal dynamics – they are not accurate enough for enabling training DRL or designing model-based methods with good performance, but nevertheless contain valuable information of building dynamics, 2) *historical data* collected from existing controllers – they may not be able to train DRL controllers with good performance due to distribution shift, but also contain useful information on building behavior, and 3) expert rules that reflect basic policies. We believe that leveraging these domain expertise can help accelerate the online RL process. In particular, our framework first learns *expert functions* from existing abstract physical models and from historical data via offline RL, and then combines those with expert rules to generate an *integrated expert function*, which will then be used to drive online RL with prior-guided learning and policy initialization from expert function distillation.

Moreover, to further improve the learned DRL-based controller's capability in keeping room temperature within the comfortable range, we propose a novel **runtime shielding framework** with an expert model. Instead of combining the temperature violation and the energy cost as the optimization objective (like during the DRL training), the framework considers the comfortable temperature range as constraints and tries to adjust the DRL-based controller's output for meeting the temperature constraints during runtime. More specifically, the expert model takes the system state as input and predicts the next-step indoor temperature and worst-case indoor temperature in the next few steps. Based on such prediction, the framework iteratively adjusts the controller output for meeting the temperature constraints. The runtime framework provides a general design for reducing temperature violation rate, where various controllers can be incorporated. In this case, when our proposed DRL-based controller is incorporated into it, significant reduction in temperature violation rate can be observed in experiments.

2.2.2 Related Works

Transfer Learning for HVAC Control: One way to speed up RL is to transfer the learned policy between different buildings. For instance, the work in [30] reduces the DRL training time by re-designing the learning objective and decomposing the neural network to a building-agnostic sub-network and a building-specific sub-network. The building-agnostic sub-network can be directly transferred from an existing DRL controller of a source building, and only the building-specific sub-network needs to be (re)-trained on the target building. This can reduce the DRL training time from months/years to weeks. The approach in [71] utilizes the direct policy transfer between different houses with the same state/action space for heat pump control in microgrids. The work in [74] applies the transfer learning to a PPO-based controller for smart home to reduce the training cost. The main limitations of these approaches is that the effectiveness of the transfer strongly relies on the similarity between the source and the target buildings. When the buildings are not similar or not operating in similar environment, the transfer may have poor performance [30].

Offline Reinforcement Learning: Another way to accelerate online RL is through *offline RL*, by leveraging historical data collected under existing control policies. Recent offline RL works focus on two aspects: offline policy optimization, and offline policy evaluation. The former aims to learn an optimal policy for maximizing a notion of cumulative reward, while the latter is intended to evaluate the accumulated reward (or the value function) of a given policy. For offline policy optimization in particular, a major challenge is that the agent cannot directly explore the environment. And the error (called extrapolation error [75]) that is caused by selected actions not contained in the historical dataset could occur and propagate during the training. This is one of the reasons that limits the effectiveness of existing offline RL approaches for building HVAC control [72]. The approaches that address this challenge mainly utilize regularization or constraint-based methods to help the policy stay near to the existing actions in the historical dataset. For instance, the batch-constrained Q-learning (BCQ) approach [75] restricts its action space to make the learned behavior similar to the actions in the historical dataset. The work in [76] penalizes divergence between the prior learned from the historical dataset and the Q-network policy using KL-control. The approach in [77] learns the policy by filtered behavioral cloning, which utilizes critic-regularized regression to filter out low-quality actions. And other related investigations can be found in [78]–[83]. From the prior experiments, we notice that not all offline RL algorithms can be chosen for building the expert function. The method like TD3+BC [83] may not always provide a good value estimation for the given states, as it only aims to make the learned policy closer to the behavior in the offline dataset and tend to overestimate the Q-value. So in this work, we use historical data as one of the expert guidance and conduct offline RL to build an expert function. We leverage the idea from [84] to estimate the value function from historical dataset because of its effectiveness, by directly setting regularization on the Q-function and generating the Q-value estimation in a conservative way to reduce overestimation.

Shielding Methods for Learning-Based Systems: Shielding methods typically first check a pre-defined shield and then adjust the control action accordingly by looking one step or a few steps ahead. For instance, to ensure safety, model predictive shielding [85], [86] leverages a backup control policy to override the learning policy when unsafe scenarios are predicted to happen. In the Simplex architecture [87], the high-assurance controller acts as a shield to the high-performance controller for improved system safety and performance. However, most shielding-based approaches rely on a fully-known environment model to synthesize a shield, which does not apply to our problem setting here where we focus on model-free building HVAC control. Moreover, shielding methods may also degrade the overall performance [88]. In this paper, we propose a novel runtime model-free shielding framework for the learned DRL-based controller. The framework does not require knowing the system dynamics, does not affect the training stage, is agnostic to the learned controller design, and as experiments show, can effectively reduce temperature violation rate while maintaining low energy cost.

2.2.3 Methodology

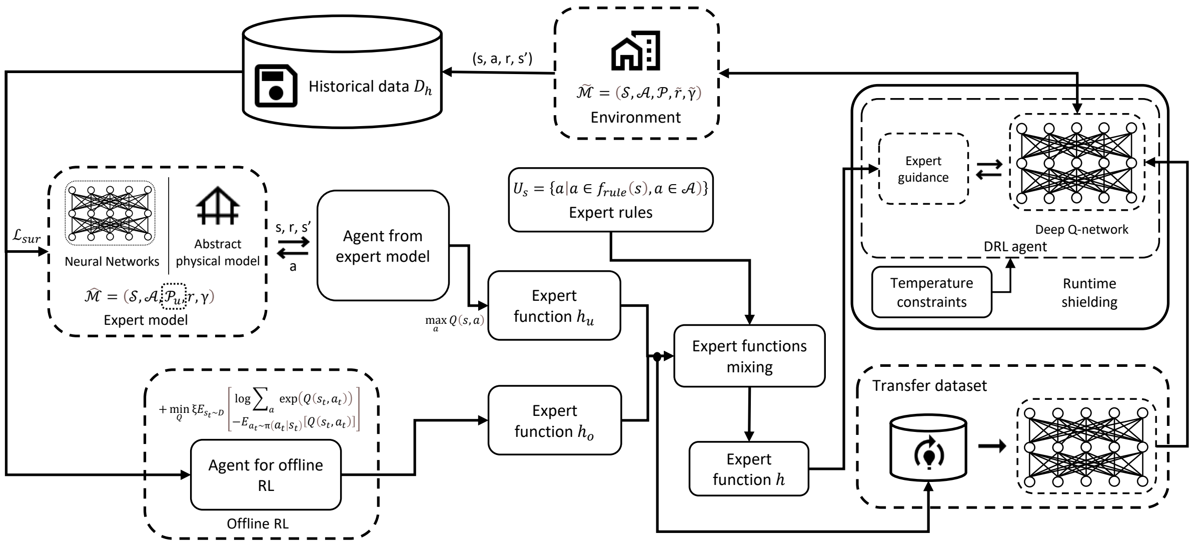


Figure 2.5: Overview of our online DRL training framework with heterogeneous expert guidances. The framework includes the following major components: (1) An expert function h_u learned from an expert model, which can be an abstract physical model or a neural network with its parameters determined from historical data. (2) Another expert function h_o learned from offline RL based on historical data. (3) An integrated expert function h generated by combining h_u and h_o as well as expert rules. (4) Application of prior-guided learning and policy initialization from expert function distillation based on h.

2.2.3 System Model

We use the building model with the fan-coil system from [70], which is extended from a single-zone commercial building with manipulable internal thermal mass. The internal air is conditioned by an idealized fan coil unit (FCU) system, and the fan airflow rate is chosen from multiple discrete levels $\{f_1, f_2, \cdots, f_m\}$ (which can be viewed as m control actions; f_1 is to turn off the cooling system, and f_m is to run it at full speed.). There are two different working modes in this system: the occupied time (daily from 7 am to 7 pm), and the unoccupied time (rest of the day). The HVAC system will run in a low-power mode during the unoccupied time for the energy-saving purpose (with the cooling system almost turned off). And the setting of comfortable temperature bound is different in these two modes. The system conducts control with a period of Δt. Each training episode contains two days of data, so there are $\frac{2880}{\Delta t}$ control steps in each episode. Other experiment-related settings can be found in experimental results section. The system state contains the following elements:

- Current physical time t,
- Indoor air temperature T_t^{in},
- Outdoor air temperature T_t^{env},
- Solar irradiance intensity q_t^{sun},
- Power consumption during the current control interval P_t,
- Outdoor air temperature forecast in the next three control steps $\{T_{t+1}^{env}, T_{t+2}^{env}, T_{t+3}^{env}\}$, and
- Solar irradiance intensity forecast in the next three control steps $\{q_{t+1}^{sun}, q_{t+2}^{sun}, q_{t+3}^{sun}\}$.

One thing to note is that we add one additional variable in the implementation to the system state design, which is the remainder after dividing the current physical time t by $24 * 60 * 60$. This is to help the RL agent figure out the time position within one day (morning, noon, afternoon, etc.), and may help it reach better performance as observed in our preliminary experiments.

2.2.3 Our Online DRL Training Framework with Heterogeneous Expert Guidances

As stated in introduction section, to accelerate online DRL for HVAC control, we propose a unified framework that leverages heterogeneous expert guidances including abstract physical models, historical data, and expert rules. Figure 2.5 shows the overview of our framework design. Specifically, the framework includes the following major components:

- An expert function h_u learned from an expert model. The expert model could be an abstract physical model developed by domain experts (commonly exists in building domain), or in case such physical model is not available, a neural network with its parameters determined from historical data (but different from offline RL; more details later).
- Another expert function h_o learned via offline RL on historical data that was collected using existing controllers.
- An integrated expert function h by combining h_u and h_o as well as expert rules.
- Application of prior-guided learning and policy initialization from expert function distillation based on h.

The detailed flow of our approach is in Algorithm 4. Next, we will first explain the underlying DRL algorithm we use, and then introduce the details of each component in our approach to improve DRL efficiency with heterogeneous expert guidance.

Underlying DRL algorithm: Similarly as in recent works [6], [30], [70], we utilize double Deep Q-learning (DDQN) [89] as the underlying DRL algorithm for our framework and also the baseline method for comparison in our experiments. We choose DDQN mainly for its convenience in leveraging the value function and the good performance it has shown for HVAC control in those recent works, but our expert-guidance approach can also be applied to improve the efficiency for other DRL algorithms.

Algorithm 4 Our Online DRL Training Framework with Heterogeneous Expert Guidances

```
1: $n_{ep1}, n_{ep2}$: number of training epochs
2: $n_{max}$: maximum training time of an epoch
3: $n_{tar}$: time interval to update target network
4: Randomly initialize Q-network $Q$
5: Learn expert function $h_u$ from expert model using Algorithm 5
6: Learn expert function $h_o$ from offline RL using Algorithm 6
7: Generate integrated expert function $h$ from $h_u$, $h_o$ and expert rules, following Equation (2.17)
8: Calculate initialization dataset $D^y_{init}$ by $h_u$, $h_o$
9: Train Q-network $Q$ by loss function $\mathcal{L}_{init}$
       $Epoch$ = 1 to $n_{ep2}$
10: Reset building environment $Env$ $t$ = 0 to $n_{max}$
11: Select action $a_t$ using epsilon-greedy
12: $s_t, s_{t+1}, r_t \leftarrow Env.execute(a_t)$
13: Update $\lambda \leftarrow \lambda_0 + (1-\lambda_0)\tanh(\alpha_\lambda((Epoch-1)*n_{max}+t))$
14: $\tilde{r}_t = r_t + (1-\lambda)\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}[h(s^{'})]$, $\tilde{\gamma} = \lambda\gamma$
15: Add transition ($s_t$, $s_{t+1}$, $a_t$, $\tilde{r}_t$) to replay buffer
16: Randomly sample a batch $B$ = ($\mathscr{S}$, $\mathscr{S}^{'}$, $\mathscr{A}$, $\mathscr{R}$) from replay buffer
17: Update Q-network $Q$ with $B$ and $\tilde{\gamma}$
18: Update target network $Q^{'}$ with interval $n_{tar}$
```

We assume that the next state of the building HVAC system only relies on the current system state, and thus HVAC control can be treated as a Markov decision process (MDP). As stated in system model section, the state $s = (t, T^{in}_t, T^{env}_t, q^{sun}_t, P_t, T^{env}_{t+1}, T^{env}_{t+2}, T^{env}_{t+3}, q^{sun}_{t+1}, q^{sun}_{t+2}, q^{sun}_{t+3})$. The discrete action space $\mathcal{A}$ contains the normalized air flow rate (0 to 1) with $m-1$ intervals. The reward is designed with consideration of indoor temperature violation and energy cost, as shown below:

$$r_t = \alpha \cdot \epsilon_t + \beta \cdot c_t, \tag{2.8}$$

where ϵ_t represents the temperature violation for the current time step, c_t is the energy cost for the current time step, and α, β are the scaling factors. More specifically, ϵ_t is defined as:

$$\epsilon_t = \max(T^{in}_i - T_{upper}, 0) + \max(T_{lower} - T^{in}_i, 0), \tag{2.9}$$

where T_{upper} is the upper bound of a given comfortable temperature range (which could be based on standards such as ASHRAE [90] or OSHA [91]) and T_{lower} is the lower bound. Moreover:

$$c_t = p_t P_t, \tag{2.10}$$

where p_t is the energy price at time t, and P_t is the power consumption during the current control interval at time t.

The goal of the DRL is to minimize total energy cost while maintaining indoor temperature within the comfortable temperature range. The loss function $\mathcal{L}_Q$ for updating the Q-network is:

$$\mathcal{L}_Q = \mathbb{E}_{(s_t,a_t,s_t^{'})\sim D}\left[(r_t + \gamma \max_{a_{t+1}} Q^{'}(s_{t+1}, a_{t+1}) - Q(s,a))^2\right], \tag{2.11}$$

where $s_t, s_{t+1} \in \mathcal{S}$, $a_t \in \mathcal{A}$, Q is the Q network and $Q^{'}$ is the target Q network. Then, the components introduced in the rest of this section will generate expert functions to provide prior guidance and policy initialization for this underlying DRL algorithm.

Learning Expert Function h_u from Expert Model: An expert function h_u can be learned through an expert model. In many cases, such expert model already exists in the form of an abstract physical model for the building thermal dynamics, e.g., an ARX or RC-networks model. While these abstract models are typically not accurate enough to enable good performance for DRL or model-based methods, they can be effectively leveraged to generate an expert function.

If an abstract physical model is not available, we can build a neural network as the expert model, with its parameters decided from historical data collected under existing control policy, as in Algorithm 5 (Line 5 in Algorithm 4) and described below.

We denote the historical dataset as D_h, with n data samples. For each data sample $(x, y) \in D_h$, let input $x = \{t, T_t^{in}, T_t^{env}, q_t^{sun}, P_t, T_{t+1}^{env}, T_{t+2}^{env}, T_{t+3}^{env}, q_{t+1}^{sun}, q_{t+2}^{sun}, q_{t+3}^{sun}, a\}$ as defined in system

Algorithm 5 Learning Expert Function from Expert Model

```
1: $n_{ep1}$: number of training epochs
2: $n_{max}$: maximum training time of an epoch
3: $n_{tar}$: time interval to update target network
4: Randomly initialize Q-network $Q_u$
5: Prepare input samples and corresponding labels $\{x^*, y^*\}$ from historical dataset $D_h$ for training an
   expert model
6: Train expert model $Env_u$ using dataset $\{x^*, y^*\}$ and loss function $\mathcal{L}_u$ $Epoch$ = 1 to $n_{ep1}$
7: Reset building environment $Env$ $t$ = 0 to $n_{max}$
8: Select action $a_t$ using epsilon-greedy
9: $s_t, s_{t+1}, r_t \leftarrow Env_u.execute(a_t)$
10: Add transition ($s_t$, $s_{t+1}$, $a_t$, $r_t$) to replay buffer $RB_h$
11: Randomly sample a batch $B$ = ($\mathscr{S}$, $\mathscr{S}^{'}$, $\mathscr{A}$, $\mathscr{R}$) from $RB_h$
12: Update Q-network $Q_u$ with $B$ and $\tilde{\gamma}$
13: Update target network $Q_u^{'}$ with interval $n_{tar}$
14: Set expert function $h_u$ using $Q_u$
```

model section and $a \in \mathcal{A}$, and let output label $y = \{T^{in}_{t+1}\}$. The neural network-based expert model consists of m_u fully-connected layers. All hidden layers are followed by a GELU activation function [92], and are sequentially connected (the detailed layer setting will be specified later in Table 2.9 of experimental results section). As different variables may not be in the same order of magnitude (e.g., t can be 1000 times larger than T^{in}_t), we normalize the input x and the output label y. The preprocessed input and output can be written as $x^* = \frac{x-x_l}{x_h-x_l}$, $y^* = \frac{y-y_l}{y_h-y_l}$, where x_h and x_l are the upper bound and lower bound of the variable x, and y_h and y_l are the upper and lower bound of the variable y. We then train the expert model with a mean square error loss function

$$\mathcal{L}_u = \parallel y^* - y^*_{pred} \parallel^2, \tag{2.12}$$

where y^*_{pred} is the network prediction for the normalized y. When we apply this expert model after model training, we obtain the prediction of y by reversing the operation of previously-mentioned normalization step. It may not be necessary to predict the entire system state, e.g., the environment temperature T_t^{out} and solar irradiance q_t^{sun} may be obtained from weather forecast.

Once we have the expert model, either in the form of an abstract physical model or a neural network, the expert function h_u can be viewed as a prior guess of the optimal value function in the building HVAC control task and can be learned via DRL. More specifically, we define an MDP problem $\hat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_u, r, \gamma)$ where the definitions of state $\mathcal{S}$, action space $\mathcal{A}$ and reward function r are the same as defined at the beginning of underlying DRL algorithm section . $\mathcal{P}_u$ is from the expert model. We then apply DDQN on $\hat{\mathcal{M}}$ and obtain a trained Q-network Q. And the expert function h_u can be set up as:

$$h_u(s) = \max_a Q(s, a), \tag{2.13}$$

where s is the state and a is the control action.

Learning Expert Function h_o from Offline RL: Another type of expert function h_o can be learned from the historical data via offline RL, as shown in Algorithm 6 (Line 6 in Algorithm 4). We leverage some of the techniques from conservative Q-learning (CQL) [84] because of its effectiveness in reducing a large number of hyper-parameters.

Algorithm 6 Learning Expert Function from Offline RL

```
1: $n_{ep1}$: number of training epochs
2: $n_{max}$: maximum training time of an epoch
3: $n_{tar}$: time interval to update target network
4: Randomly initialize Q-network $Q_o$
        $Epoch$ = 1 to $n_{ep1}$ $t$ = 0 to $n_{max}$
5: Randomly sample a batch $B$ = ($\mathscr{S}$, $\mathscr{S}'$, $\mathscr{A}$, $\mathscr{R}$) from $D_h$
6: Update Q-network $Q_o$ with $B$ and $\tilde{\gamma}$ following Equation 2.15
7: Update target network $Q_o'$ with interval $n_{tar}$
8: Set expert function $h_o$ using the learned Q-networks $Q_o$
```

First, we build an offline RL model based on DDQN, but with different Q-network updating rules as the DRL presented in the beginning of underlying DRL algorithm section. Compared with

Equation (2.11), we add an extra regularization term:

$$\mathcal{L}_{reg} = \min_{Q} \mathbb{E}_{s_t \sim D} \left[\log \sum_{a_t} \exp(Q(s_t, a_t)) - \mathbb{E}_{a_t \sim \pi(a_t|s_t)} \left[Q(s_t, a_t) \right] \right], \tag{2.14}$$

where $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$. Q is the Q-network, and D is the dataset produced by the behaviour policy π. In the equation, the first part $\log \sum_{a_t} \exp(Q(s_t, a_t))$ describes a penalty term for minimizing the Q-value of the action produced by current policy on the states in the historical dataset. It helps learn a smaller and more conservative Q-value estimator. The second term $-\mathbb{E}_{a_t \sim \pi(a_t|s_t)}[Q(s_t, a_t)]$ counts average Q-value in the state-action pairs in the historical dataset and maximizes it to push the current learned policy closer to the behavior policy in the historical dataset.

Then the policy updating is changed as follows:

$$\mathcal{L}_{off} = \frac{1}{2} \mathbb{E}_{(s_t, a_t, s_t^{'}) \sim D} \left[(r_t + \gamma \max_{a_{t+1}} Q^{'}(s_{t+1}, a_{t+1}) - Q(s_t, a_t))^2 \right] + \xi \mathcal{L}_{reg}, \tag{2.15}$$

where $s_t, s_{t+1} \in \mathcal{S}$, ξ is a mixing coefficient, and $Q^{'}$ is the target Q-network. With enough training iterations, the offline RL agent can provide a good expert function h_o following the same procedure as in Equation (2.13).

Note that we observe that not all offline RL algorithms can be a suitable choice for our framework. For example, approaches like TD3+BC [83] may not always provide a good value estimation for the given states. We suspect that this may be due to two factors. One is related to the reward design, as the value function estimation in some offline RL algorithms is sensitive to the scale of the accumulated reward. The other is that because algorithms like TD3+BC only add regularization on the actor updating and do not set constraints on the Q function, which could enlarge the error in estimating the (Q-)value function when combined with possible numerical issues.

Generating Integrated Expert Function h from h_u, h_o and Expert Rules: The expert function

h_u learned from the expert model and the expert function h_o learned via offline RL tend to perform differently because of the complexity of the system dynamic and the sufficiency of the data. Moreover, the accuracy of their Q-value estimation can vary at different states depending on the data distribution within the historical dataset. Thus, it is a natural thought to form an ensemble of the two. And the ensemble of multiple expert functions calculated in different ways can further reduce the overestimation of Q-values through a conservative way, which we will introduce in this section later.

To begin with, after having h_u and h_o, we can combine them with *expert rules* to generate an integrated expert function h. The expert rules are often set by domain experts or building operators based on past experience and domain expertise. They do not provide an optimized control action for a given state, but instead offer suggestions that could be viewed as guidance or soft constraints – e.g., not turning on the cooling system when the indoor temperature is below the lower bound of the comfortable temperature range by certain threshold. Formally, we define that the expert rules f_{rule} can generate an action candidate set $\mathbf{U}_s$ for each state:

$$\mathbf{U}_s = \{a | a \in f_{rule}(s), a \in \mathcal{A})\}. \tag{2.16}$$

We can then generated an integrated expert function h based on $\mathbf{U}_s$, h_u and h_o (Line 7 in Algorithm 4). Specifically, we apply a pessimistic ensemble strategy for selecting the value function estimation among different expert functions, and only choose corresponding actions from the expert rules' action candidate set $\mathbf{U}_s$. Thus, the integrated expert function h can be formulated as:

$$h(s) = \min_i (\max_{a \in \mathbf{U}_s} Q_i(s, a)), \tag{2.17}$$

where Q_i is the Q-value estimation from expert functions i. Note that this is a *general formulation* that can unify multiple expert functions – e.g., we may have more than one abstract physical models that provide multiple h_u expert functions.

Prior-guided Learning: Once we have the integrated expert function h, we can use it to guide the underlying DRL with prior-guided learning. There are several algorithms that could guide online RL with a single prior policy, such as HuRL [93] and JSRL [94]. Our framework is flexible in choosing those and we select HuRL [93] in our implementation. In the original HuRL, the Q-value estimation in the RL agent is guided by a simple heuristic function that is learned from the Monte-Carlo regression. In our work, we instead leverage the integrated expert function h from above. By dynamically changing a mixing coefficient λ that controls the trade-off between the bias from the expert function h and the complexity of a reshaped MDP, we are able to accelerate the DRL training with a shortened MDP horizon. Specifically, given the state space $\mathcal{S}$, action space $\mathcal{A}$, reward function r that are mentioned at the beginning of underlying DRL algorithm section, as well as the transition dynamics of the building HVAC system $\mathcal{P}$ and a discount factor γ, we consider an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$. We use the learned integrated expert function h as a prior guess for the optimal value function of $\mathcal{M}$. Thus our online DRL can be described as a reshaped MDP $\tilde{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \tilde{r}, \tilde{\gamma})$, where λ is a mixing coefficient,

$$\tilde{r} = r + (1-\lambda)\gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)}[h(s^{'})], \tag{2.18}$$

and

$$\tilde{\gamma} = \lambda\gamma, \tag{2.19}$$

which is shown at Line 14 in Algorithm 4.

Policy Initialization from Expert Function Distillation: In the above section, we use the integrated expert function h to reshape the reward function and shorten the MDP horizon. In addition, we can also speed up the DRL training through better initialization, by leveraging the expert functions for determining the initial policy (Lines 8 and 9 in Algorithm 4).

Specifically, we initialize the deep Q-network through knowledge distillation [95] on the expert functions. The first step is to extract the knowledge from multiple expert functions (h_u and h_o in our case) to a dataset D_{init}. We set the input dataset as D^x_{init} and the corresponding label set as D^y_{init}. In setting D^x_{init}, we utilize all the unlabeled historical data, which only contain the system state. And the corresponding labels are calculated in a way that is similar to the strategy introduced earlier for integrating expert functions. That is, suppose we have n_h expert functions, then

$$D^y_{init} = \{y|y = (q_1, q_2, \cdots, q_m)\}, \tag{2.20}$$

$$q_j = \min_i(Q_i(s, f_j)), \tag{2.21}$$

where $s \in D^x_{init}, j \in [1 \cdots m], i \in [1 \cdots n_h]$. As the expert functions we utilize are not as accurate as of the optimal (Q-) value function, we further add two mixing coefficients λ^α_{init}, λ^β_{init} for balancing the relative size of the Q value from different actions. So the new definition of D^y_{init} is

$$D^y_{init} = \{y|y = (\frac{q_1 + (\lambda^\alpha_{init} - 1)\mu_q}{\lambda^\alpha_{init}\lambda^\beta_{init}}, \frac{q_2 + (\lambda^\alpha_{init} - 1)\mu_q}{\lambda^\alpha_{init}\lambda^\beta_{init}}, \cdots, \frac{q_m + (\lambda^\alpha_{init} - 1)\mu_q}{\lambda^\alpha_{init}\lambda^\beta_{init}})\}, \quad \mu_q = \frac{\sum_{j=1}^m q_j}{m}, \tag{2.22}$$

where the definition of $q_j (j \in [1 \cdots m])$ remains the same. Then the next step is to train the deep Q-network of our DRL agent by using the obtained dataset D_{init}. As we consider a regression task, we apply the mean square error as the loss function

$$\mathcal{L}_{init} = \parallel y - y_{pred} \parallel^2, y \in D^y_{init}, \tag{2.23}$$

where y_{pred} is the deep Q-network prediction. We obtain the network weight initialization by training for n_{init} epochs. Moreover, with such policy initialization, we can use a smaller learning rate to tune the deep Q-network in the later DRL stages.

2.2.3 *Runtime Shielding Framework*

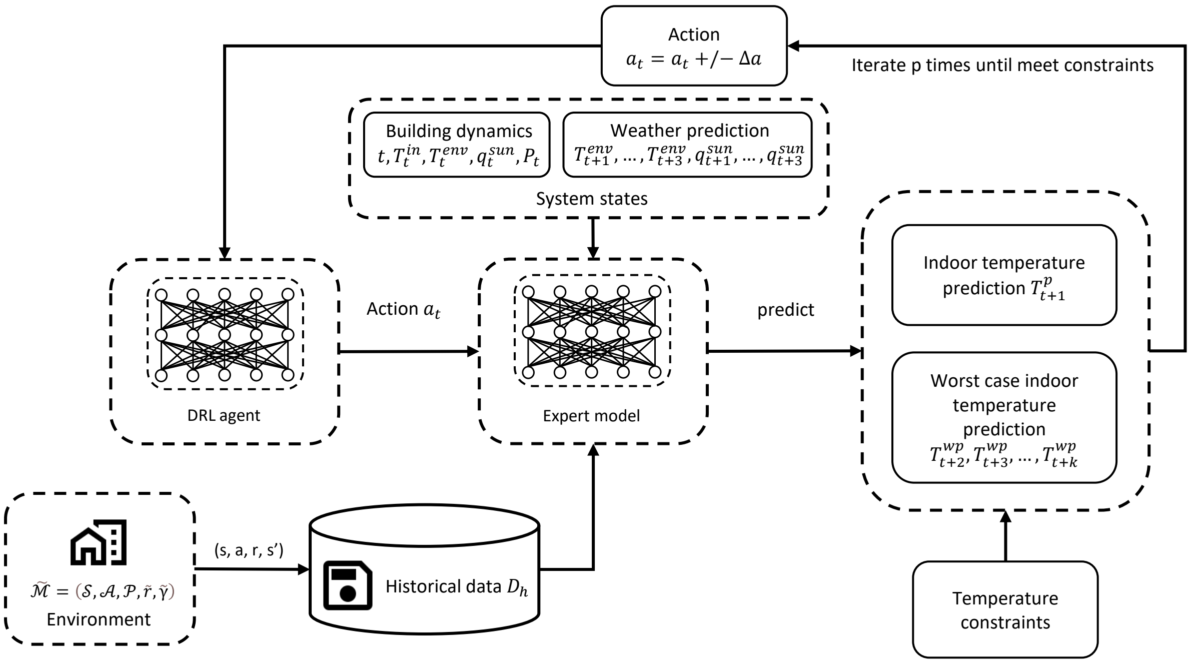

Action
$a_t = a_t +/- \Delta a$
Iterate p times until meet constraints
Building dynamics
$t, T_t^{in}, T_t^{env}, q_t^{sun}, P_t$
Weather prediction
$T_{t+1}^{env}, \dots, T_{t+3}^{env}, q_{t+1}^{sun}, \dots, q_{t+3}^{sun}$
System states
Action a_t
predict
DRL agent
Expert model
Indoor temperature prediction T_{t+1}^{p}
Worst case indoor temperature prediction
$T_{t+2}^{wp}, T_{t+3}^{wp}, \dots, T_{t+k}^{wp}$
$\widetilde{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \tilde{r}, \tilde{\gamma})$
Environment
(s, a, r, s')
Historical data D_h
Temperature constraints

Figure 2.6: Overview of our runtime shielding framework for reducing temperature violation rate. The framework includes two major components: 1) the learned DRL agent (by our online DRL training framework as introduced earlier) that produces the control action based on the current system state, and 2) an expert model for predicting indoor temperature in future steps, based on a neural network with its parameters determined from historical data. More specifically, the expert model takes the current system states and proposed action from the DRL agent as input, and predicts the indoor temperature for the next step and the worse-case indoor temperature for the next few steps. Based on such predictions, the control action may be adjusted iteratively to meet the temperature constraints.

As shown in the previous sections, while the training of our DRL agent considers both the temperature violation and the energy cost in the reward function design, there is no explicit enforcing of the constraints on comfortable temperature range. Similarly for many other learning-based (and model-based) controllers, there is no explicit enforcing of the temperature constraints on the control actions, which may lead to constant temperature violations. Thus, in this work, we propose a novel runtime shielding framework to help HVAC controllers meet temperature constraints. The framework does not affect the controller training process and is agnostic to the controller design.

Figure 2.6 shows the overview of our runtime shielding framework, which integrates the HVAC controller – in this case, the DRL-based controller trained by our proposed online framework under heterogeneous expert guidances – with an expert model that predicts future indoor temperature based on the system states and the control input. The expert model is trained from the historical data collected from the building environment, similarly as the one used in our online training framework but with different goal and output. More specifically, during runtime, the learned DRL agent proposes a control action a_t for the current time t. The expert model for temperature prediction takes the proposed control action a_t from the DRL agent and the system states s_t as input, and outputs the indoor temperature prediction that includes not only the temperature prediction for the next time step T^p_{t+1} but also the *worst-case* temperature prediction from time $t+2$ to $t+k$, named as $T^{wp}_{t+2}, \ldots, T^{wp}_{t+k}$.

In particular, at time $t+i$ $(2 \leq i \leq k)$, the expert model predicts the worst cases regarding both the temperature upper bound and the temperature lower bound as follows: (1) Based on the predicted temperature at time $t+i-1$ and the worst-case control action for temperature lower bound at time $t+i-1$, which is $a^{wp,l}_{t+i-1} = 0$, the expert model will predict the indoor temperature at time $t+i$, named $T^{wp,l}_{t+i}$. This is a worst-case prediction of the temperature lower bound, i.e., $a_t = \max(a_t - \Delta a, 0)$, if $T^{wp,l}_{t+i} < T_{lower}$. (2) On the other hand, based on the predicted temperature at time $t+i-1$ and the worst-case control action for temperature upper bound at time $t+i-1$, which is $a^{wp,u}_{t+i-1} = m-1$, the expert model will predict the indoor temperature at time $t+i$, named $T^{wp,u}_{t+i}$. This is the worst-case prediction of the temperature upper bound, i.e., $a_t = \min(a_t + \Delta a, m-1)$,

Parameter	Value	Parameter	Value
Expert-model	$[len(s \in \mathcal{S})$, 256,256,256, 256,256,256,2]	Deep Q-network	$[len(s \in \mathcal{S})$, 256, 256, 256, 256, 51]
m	51	Δt	15 mins
γ	0.99	α	1.0
T_{lower} (occupied)	22 °C	T_{upper} (occupied)	26 °C
T_{lower} (unoccupied)	12 °C	T_{upper} (unoccupied)	30 °C
β	100.0	m_u	7
ξ	1.0	n	5760

Table 2.9: Hyper-parameters used in our experiments.

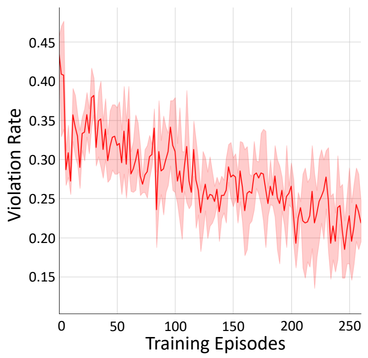
if $T^{wp,u}_{t+i} > T_{upper}$. Then, the comfortable temperature range will serve as the constraints against which these temperature predictions are checked. If the predictions are out of the range, the current proposed control action a_t will be iteratively adjusted until the temperature constraints are met or the iteration number reaches p. Moreover, if such change for the proposed control action based on the prediction of time $t+i$ makes the previous temperature predictions (i.e., $T^{p}_{t+1}, T^{wp}_{t+2}, \ldots, T^{wp}_{t+i-1}$) violate the temperature constraints, we will discard the result, stop next worst-case predictions, and use the results from time $t + i - 1$.

2.2.4 Experimental Results

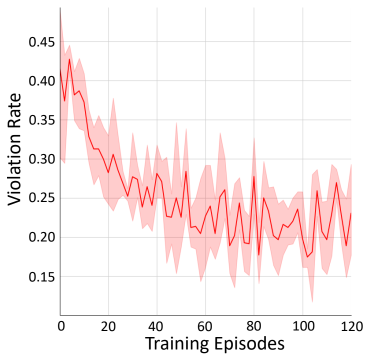
2.2.4 Experiment Settings

We conduct our experiments on a Ubuntu 20.04 OS server equipped with NVIDIA RTX A5000 GPU cards. Docker [96] is utilized for the environment configuration, with Python 3.7.9 and learning framework Pytorch 1.9.0. All neural networks are optimized through the Adam optimizer [65].

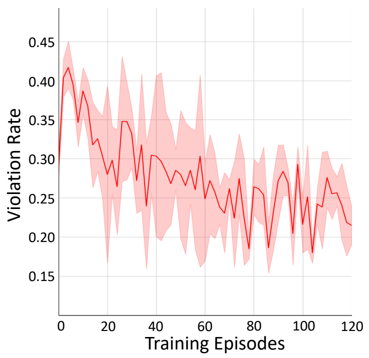
We use the building simulation tool in [70] to simulate the behavior of single-zone commercial buildings, with an OpenAI-Gym [97] interface. We model two buildings as defined in the Building



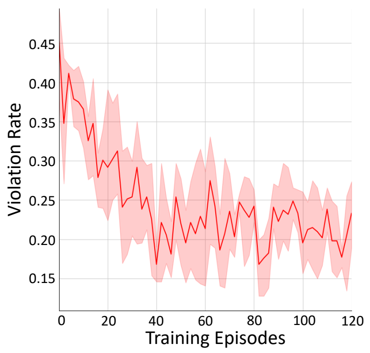
((a)) DDQN



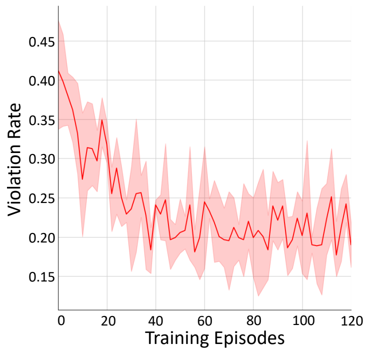
((b)) DDQN+Expert Model



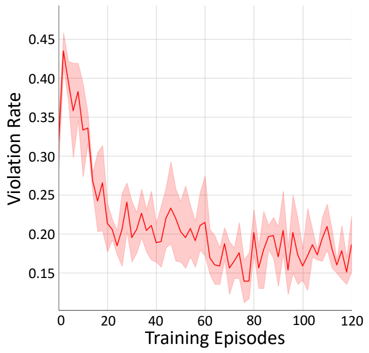
((c)) DDQN+Offline RL



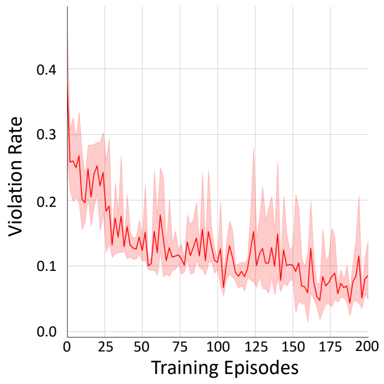
((d)) DDQN+Expert Model+Offline RL



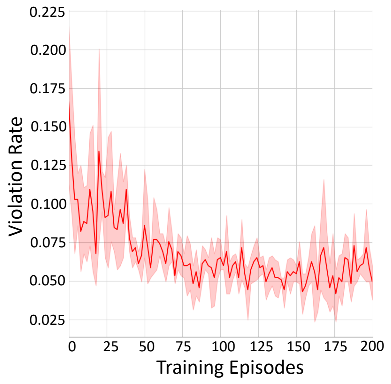
((e)) DDQN+Expert Model+Offline RL+Expert Rules



((f)) DDQN+Expert Model+Offline RL+Expert Rules+Init



((g)) DDQN, heavyweight building



((h)) DDQN+Expert Model+Offline RL+Expert Rules+Init, heavyweight building

Figure 2.7: Figure 2.7(a) to Figure 2.7(f) show the comparison between our online DRL training framework (in different settings with various techniques included) and the standard DDQN method on the lightweight building. The x-axis shows the training episodes. The y-axis shows the temperature violation rate. Figure 2.7(a) shows the training process under the standard DDQN method. About 212 episodes are needed to reach a violation rate of 0.2. Figure 2.7(b), Figure 2.7(c), Figure 2.7(d), and Figure 2.7(e) show the results when we gradually add an expert model that generates expert function h_u, offline RL that generates expert function h_o, an expert rule, and policy initialization based on expert functions, respectively. And we can observe the improvement on the required episodes step by step. Figure 2.7(f) shows the training process when we apply all of our techniques with only 24 episodes are needed to reach the violation rate of 0.2, an $8.8X$ improvement over standard DDQN. Then Figure 2.7(g) and Figure 2.7(h) show the comparison between our approach with all techniques included (right) and the standard DDQN baseline (left) on the heavyweight building with larger thermal capacity.

Energy Simulation Test validation suite [98]: one is with a lightweight construction (known as Case600FF) and the other is with a heavyweight construction (known as case900FF). Both buildings have the same model settings except that the wall and floor construction have either light or heavy materials. The floor dimensions are $6m$-by-$8m$ and the floor-to-ceiling height is $2.7m$. There are four exterior walls facing the cardinal directions and a flat roof. The walls facing east-west have the short dimension. The south wall contains two windows, each $3m$ wide and $2m$ tall. The use of the building is assumed to be a two-person office with a light load density. The lightweight building is assumed to be located at Riverside, California, USA, and the heavyweight building is assumed to be located at Chicago, Illinois, USA. The weather data for different locations are obtained from the Typical Meteorological Year 3 database [68]. In addition, the various parameters and hyper-parameters mentioned in the previous sections are listed in Table 2.9.

2.2.4 Evaluation of Our Online DRL Training Framework

We first apply our proposed online RL framework with heterogeneous expert guidances to building HVAC control and demonstrate its effectiveness in accelerating the DRL training, in particular for the standard DDQN algorithm. We repeat each experiment 4 times and show the average results.

Comparison with Standard DDQN on Training Efficiency: Figure 2.7 demonstrates the temperature violation rate of the trained controller under different approaches for the lightweight building with weather data from Riverside. Temperature violation rate is one of the main objectives for DRL. It is defined as the percentage of the time the indoor temperature is outside of the comfortable temperature zone, similarly as used in [6], [30], [35], [70].

Figure 2.7(a) shows the training process of the standard DDQN, and the model needs **about 212 episodes to reach a violation rate at around 20%** for this building from [70] (20% may seem high, but it is due to the limitation of this particular building and its cooling-only HVAC system; more explanation on this later with Figure 2.9). Figure 2.7(b) shows the training process when we add a neural network-based expert model that generates the expert function h_u. About 68 episodes are needed to reach the same violation rate. Figure 2.7(c) shows the training process when we add offline RL that generates the expert function h_o, and about 78 episodes are needed to reach the violation rate of 20%. Figure 2.7(d) shows the results when we apply both expert functions h_u and h_o, but without the expert rules. We can see that about 40 episodes are needed. Figure 2.7(e) shows the results when we integrate the two expert functions h_u and h_o, as well as an expert rule f using the method introduced in Equation 2.17 . f is defined as follows: when the indoor temperature is below 22°C, the control action is suggested to be set within the set of $\{f_0, f_1, f_2, f_3\}$; if the indoor temperature is above 27°C, the control action is suggested to be set within the set of $\{f_{m-3}, f_{m-2}, f_{m-1}, f_m\}$. We can see that the number of episodes needed is about 36. Finally, Figure 2.7(f) shows the training process when we apply all of our proposed techniques, including integrating the expert functions from expert model and offline RL as well as the expert rules, using the integrated expert function to guide DRL training, and conducting policy initialization with the expert functions. We can see that **now only 24 episodes are needed to reach the same violation rate as the standard DDQN, an 8.8X reduction in training time**. Table 2.10 summarizes the above number of episodes required to reach the violation rate of 0.2 for the standard DDQN baseline and our approach with various techniques included.

For further evaluation, we also conduct experiments on the heavyweight building with weather data from Chicago. In this set of experiments, the major change of the parameters is that the scaling factor β is set to 1.0 in Equation (2.5). This is because that the average energy consumption of this HVAC system is much higher than that of the previous building, and we need to re-balance the energy cost and the temperature violation in the reward design. Figure 2.7(g) and Figure 2.7(h) shows the comparison between our approach and the standard DDQN. And the experiments show

Method	Number of Episodes
DDQN	212
DDQN+Expert Model	68
DDQN+Offline RL	78
DDQN+Expert Model+Offline RL	40
DDQN+Expert Model+Offline RL +Expert Rules	36
DDQN+Expert Model+Offline RL +Expert Rules+Init	24

Table 2.10: Number of episodes required to reach the violation rate of 0.2 for the standard DDQN baseline and our online DRL training framework with various techniques included (the last line being our approach with all techniques in Algorithm 4).

that the number of episodes needed to reach a violation rate of 5% is reduced from 160 to 80. The improvement, while still significant, is much less than the lightweight building. We suspect that this may be due to the quality of the historical data and plan to investigate it further in future work.

Energy Cost and Other Details: Besides temperature violation rate and the number of episodes for reaching the goal of violation rate below 0.2 (i.e., training efficiency), we also assess the energy cost of the learned controllers during our experiments. We observed that different methods, including the standard DDQN baseline and our approach with various techniques included, achieve very similar energy cost for the learned controllers – in fact within 1% for both the lightweight building and the heavyweight building we tested.

Figure 2.8 shows the normalized energy cost of our approach with all techniques included for the lightweight building with weather data from Riverside. We can observe that the energy cost quickly decreases to a lower value within 5 to 10 epochs and slightly fluctuates in the later training epochs.

Figure 2.9 illustrates the building temperature over 2 days, under the controller learned with our approach with all techniques included, for the lightweight building with weather data from

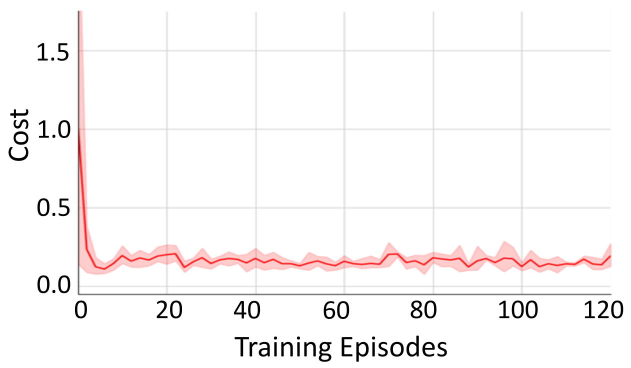


Figure 2.8: Normalized energy cost during training for our approach with all techniques included for the lightweight building with weather data from Riverside.

Riverside. We can see that the temperature violation rate is around 20%. It is relatively high because some violations are very hard to avoid for this particular building. Specifically, the HVAC system is set to only work during the occupied hours (from 7am to 7pm) and the comfortable temperature range is much more strict during that time (22°C to 26°C) compared to during the unoccupied time (12°C to 30°C) [70]. This makes it almost impossible to meet the comfortable temperature range early in the morning since the HVAC system only provides cooling. We can see that after the early morning hours, the temperature is controlled well within the comfortable range by our controller.

2.2.4 Ablation Studies

Impact of the Historical Data Quantity: We are interested in knowing how the quantity of the historical data may affect the performance of our approach. We conduct a series of experiments that have the quantity of the historical data chosen from $\{5760, 2880, 1440, 720\}$ (i.e., from 2 months of data to 7.5 days of data). The results are shown in Table 2.11. We can observe that the training becomes faster as the quantity of the historical data becomes larger, as what we would expect.

Impact of the Control Quality of Historical Data: We also study the performance of our ap-

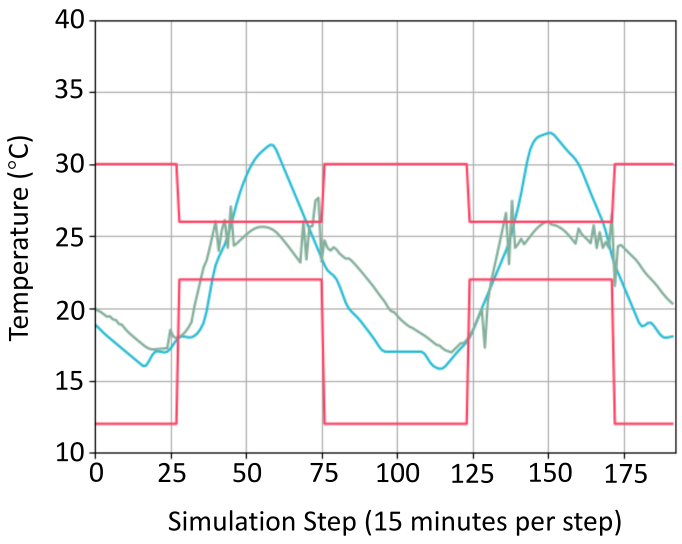


Figure 2.9: An illustration of the lightweight building temperature over 2 days under the controller learned from our approach with all techniques included. The red lines bound the comfortable temperature range. The blue line is the outdoor temperature in Riverside, CA. The green line is the indoor temperature under the learned controller.

#Samples	720	1440	2880	5760
#Episodes	116	78	62	24

Table 2.11: The number of epochs needed by our approach (with all techniques included) for reaching the violation rate of 20% for the lightweight building, under different quantity of the historical data.

proach under different levels of control quality of the historical data. Previously we directly use the historical data collected from an existing controller on the target building. To study different control quality of such historical data, we choose to take random actions with a probability of p – intuitively, higher p values implies more random control and hence worse quality. Table 2.12 shows the results. Our approach performs better with a smaller p, i.e., when our approach learns from historical data based on more reasonable control actions.

The Usage of Abstract Phyiscal Model: In addition, we also try to utilize an abstract physical

p	1.0	0.8	0.4	0.2	0.0
#Episodes	110	104	88	60	24

Table 2.12: The number of epochs needed by our approach (with all techniques included) for reaching the violation rate of 20% for the lightweight building, under different control quality of the historical data.

model, i.e., the ARX model from [35], as the expert model to generate h_u, instead of learning a neural network. The training process is shown in Figure 2.10. About 64 episodes are needed to reach the same violation rate, more than the case where the expert model is a neural network learned from historical data. We think that this is due to the simplicity of the ARX model, and plan to investigate the performance of other abstract physical models in future. Nevertheless, it still provides considerable improvement over the standard DDQN.

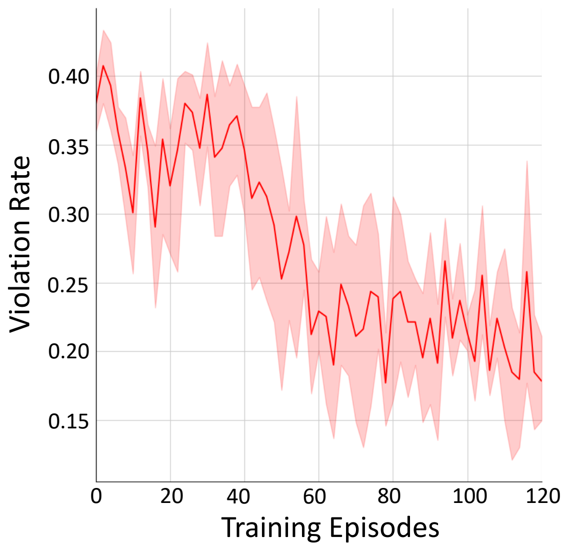


Figure 2.10: Training result for the lightweight building when the expert model in our approach (with all techniques included) is constructed from an abstract physical model.

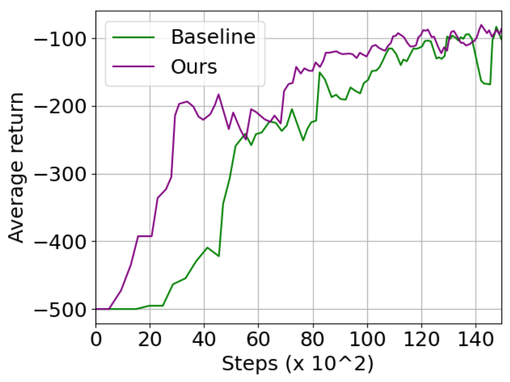
2.2.4 Evaluation of Our Runtime Shielding Framework

For evaluating our proposed runtime shielding framework, in particular its capability in further reducing the temperature violation rate for our learned DDQN-based DRL controller, we conduct experiments on the heavyweight building. The expert model predicts the indoor temperature for the next step and the worst-case indoor temperature for another step ahead. We consider two DDQN agents – agent 1 is coarsely trained and agent 2 is the final model after additional training (the same one shown in Figure 2.7(h)).

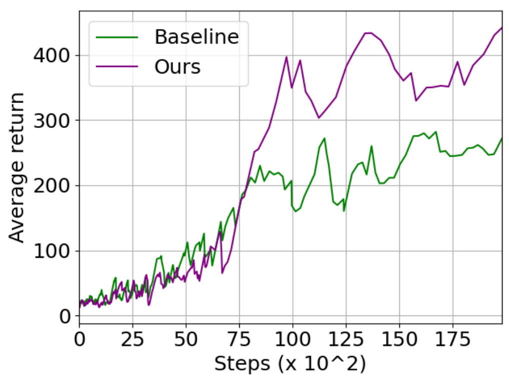
From the results shown in Table 2.13, we can observe that while the temperature violation rate of agent 1 is quite high, our runtime shielding framework can decrease it by more than $3X$ (from 26.56% to 7.81%), with only slight increase in energy cost. For agent 2, which has a much lower temperature violation rate than agent 1 given the additional training, our runtime shielding framework can still reduce the violation rate substantially, from 4.17% to 3.65%, with slight reduction on energy cost as well. These results demonstrate the **effectiveness of our shielding framework in reducing temperature violation rate while keeping similar energy cost**, for controllers with varying qualities. Note that further improvement on temperature violation rate for this particular system is challenging, due to the limitation on its cooling-only HVAC system (i.e., there is always a short period in the early morning when the indoor temperature is below the desired lower bound and the HVAC system has just started, as shown in Figure 2.9).

2.2.4 Experiments in Other Domains

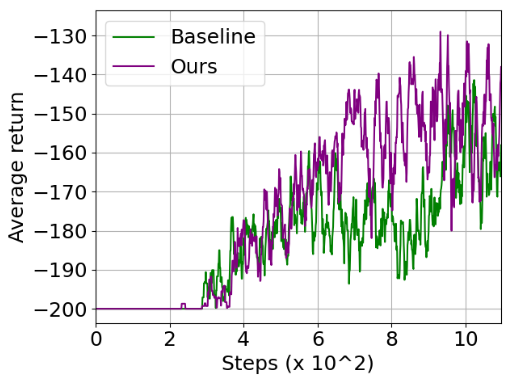
We believe that our approach of leveraging existing domain expertise in DRL training may be extended to other domains of cyber-physical systems. Thus, we conduct initial exploration outside of the building domain, on a few examples from the Gym [99] environment, to assess our approach's general applicability. Figure 2.11 shows the comparison between our approach and the standard



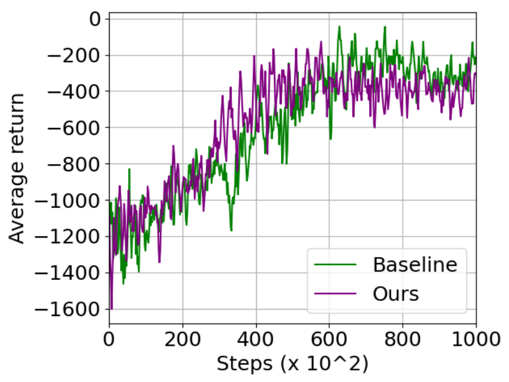
((a)) Acrobot



((b)) CartPole



((c)) MountainCar



((d)) Pendulum

Figure 2.11: Results on examples from the Gym environments.

Method	Temperature Violation Rate (%)	Energy Cost
DDQN agent 1	26.56	1.60
DDQN agent 1 + Runtime Shielding	7.81	1.68
DDQN agent 2	4.17	2.07
DDQN agent 2 + Runtime Shielding	3.65	1.94

Table 2.13: Comparison between two DDQN agents (trained by our online DRL training framework to different degrees) and when they are incorporated into our runtime shielding framework, in both temperature violation rate and energy cost.

DDQN baseline. We observe that our approach is able to significantly improve the learning efficiency and/or performance on some examples (particularly CartPole) but not others (Pendulum in particular). We think that in these cases the final results can be significantly affected by the quality of the generated expert functions. Apart from the factor of offline data quality, another key factor is the difficulty of constructing a good expert model from the historical data. Compared with the accuracy of predicted system states, the accuracy of the terminate condition in each step can have a larger influence in some tasks. However, the terminate condition is always fixed in building HVAC control (run for certain steps) and we can even utilize the prior knowledge from the building domain combined with the offline data to help construct the expert model. These factors make the building HVAC control an idea application for our approach.

2.3 Learning-based framework for sensor fault-tolerant building HVAC control with model-assisted learning

2.3.1 Background

In the last two sections, we mention that the model-free building HVAC control may facing data insufficiency and suffer from long training time. On the other side, imperfect sensing data may

Section 2.3 is based on our work published at [100].

also affect the building's performance and reliability. To be specific, in HVAC 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 [101]. 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) [102], 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 [103]. 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 [104] 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. [105] 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. [106] 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 [29] 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. [107] 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, [29] 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 [104], [105], [108], 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 section, 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 [40], [109]. They require much less effort to develop than the accurate physical models (e.g., those used in EnergyPlus [110]). 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.

2.3.2 Related Works

2.3.2 Addressing Sensor Faults in Buildings

There has been a number of works in the literature addressing sensor faults in buildings. In [111], a fault detection method based on correlation analysis was proposed for detecting sensor bias or complete failure. [112] proposed a neural network-based strategy with clustering analysis to detect sensor faults in the HVAC system and diagnose the sources. [113] 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 [114]–[119]. 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 [29], [104]–[108]. For instance, Gunes et. al. [106] followed the model-based design paradigm and used rule-based methods to mitigate the negative effect of specific sensor faults. Papadopoulos et. al. [29] 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 [108] 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.

2.3.2 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 [120], [121] and semi-supervised learning [14], [122], [123] 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 [40], [109] to reduce the data needed for training. This approach is in principle related to model distillation techniques [95], [124] 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 find-tuning.

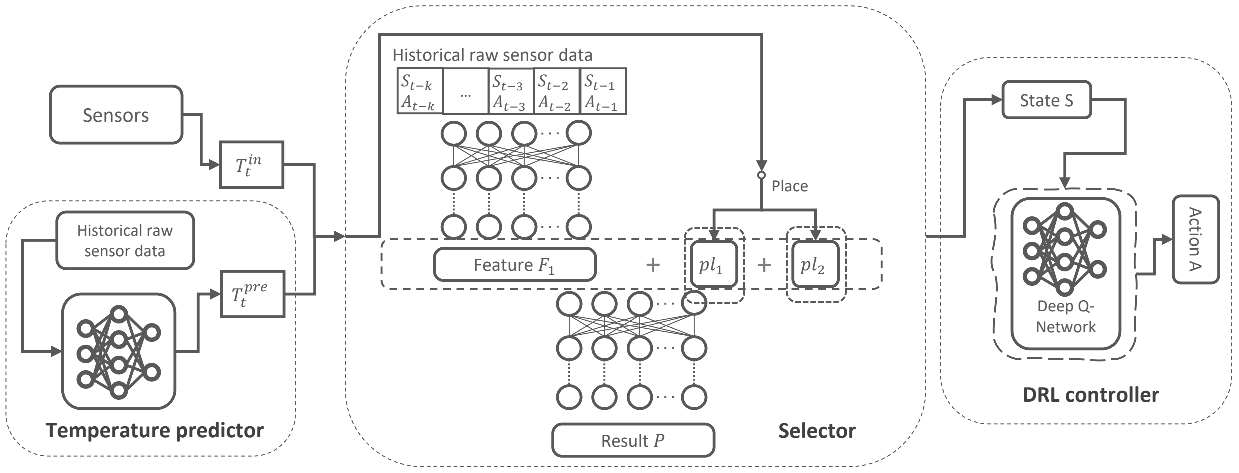


Figure 2.12: Overview of our sensor fault-tolerant framework for building HAVC system. There are three main components: two modules providing indoor temperature proposals on the left, a selector in the middle, a DQN-based HVAC controller on the right. The temperature proposals consist of the raw sensor 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.

2.3.3 Methodology

2.3.3 System Model

We adopted a multi-zone building model with the fan-coil system from [6], [30], 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 $\{f_1, f_2, \cdots, f_m\}$, and corresponding to m control actions a_i for each zone i. With all n thermal zones, the control action set is denoted as $A = \{a_1, a_2, \cdots, a_n\}$. 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 $T_t^{in} = \{T_t^{in(i)} | i \in$

$1 \cdots n\}$. 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).

2.3.3 Sensor Fault-Tolerant DRL Framework

Fig. 2.12 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. This predictor and selector design enable us to leverage the raw sensor readings when they are not faulty, which is especially useful when the temperature predictor does not provide accurate estimation because of the training data availability (As we will introduce in the later section, the selector has more augmented training data than the predictor to ensure its accuracy). Note that all the modules are trained individually and assembled into the framework after training.

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 2.14 of Section 2.3.4). 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 period of time has not been polluted by sensors faults or attacks. And we get a (state, action) sequence from (S_1, A_1) to (S_L, A_L). For the convenience of supervised training, we select data sequences

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

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 $k+1, k+v+1, k+2v+1, \cdots$. 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 $\mathcal{L}_{pre}$ as

$$\mathcal{L}_{pre} = \parallel (T_t^{pre} + T_{ofs}^{pre}) - T_t^{in} \parallel^2, \tag{2.24}$$

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 2.14 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 slightly 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, for enhancing the robustness of the temperature prediction, in the historical system states, we utilize the historical indoor temperature under the independent and identically distributed (IID) faults with occurring probability P_{pre}. IID faults here mean that the fault can happen at each individual simulation step with probability P_{pre}. 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, to replace the original indoor sensor temperature reading. And the temperature predictor takes benefit from randomized faults in the reading, which leads to a more robust output. Note that we set P_{pre} to a small value as the ability of the neural network for 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.

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}), $\cdots$, (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. The motivation of the network design here is that the historical states are the sequential data that have temporal locality, thus we choose one-dimensional convolutional layers to extract their feature. As the temperature proposals are another type of data, we concatenated the previous feature with these temperature proposals and fused them using the fully connected layers. As shown in Fig. 2.12, 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 RuLU 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 2.3.3.2. 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 2.14 later.

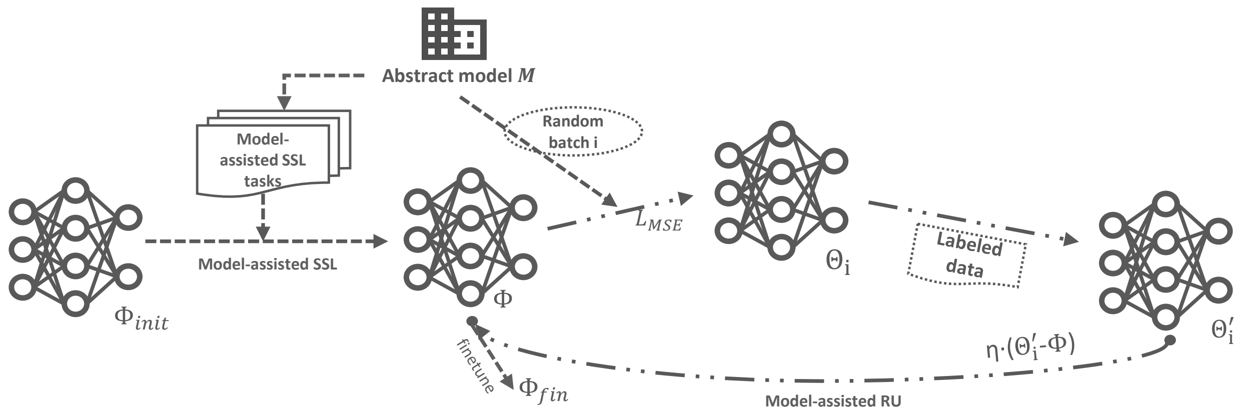


Figure 2.13: Overview of our model-assisted learning for training with a limited amount of labeled data and an abstract physical model, where the algorithm consists of two stages – model-assisted self-supervised learning (model-assisted SSL) and model-assisted redirected updating (model-assisted RU). 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.

DRL-based Controller for Building HVAC System:

Because the thermal zone temperature in the next time step only relies on the observation of 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 $T_{t+1}^{out}, \cdots, T_{t+3}^{out}$, $Sun_{t+1}, \cdots, Sun_{t+3}$, 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

$$R_t = \alpha \cdot R_c + \beta \cdot R_v \tag{2.25}$$

$$R_c = -cost(t-1, A_{t-1}) \tag{2.26}$$

$$R_v = -\sum_{i=1}^{n} \max(T_l - T_t^{in(i)}, 0) + \max(T_t^{in(i)} - T_u, 0) \tag{2.27}$$

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

$$\begin{aligned} Q_{t+1}(S_t^{DRL}, A_t) &= Q_t(S_t^{DRL}, A_t) + \eta_0(R_{t+1} \\ &+ \gamma \max_{A_{t+1}} Q_t(S_{t+1}^{DRL}, A_{t+1}) - Q_t(S_t^{DRL}, A_t))) \end{aligned} \tag{2.28}$$

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

2.3.3 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 [57], [58], 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 2.3.3.2, 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. 2.13.

Abstract Physical Model: Here we introduce the abstract physical model we used in experiments for model-assisted learning. The mass and energy conservation law for a building zone is presented in Equation (2.29), 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 in Equation (2.30). 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.

$$mC_p \frac{dT}{dt} = \dot{m}C_p(T^{sa} - T) + \dot{q}^l \tag{2.29}$$

$$\hat{T}_{t+1} = \alpha T_t + \beta \dot{m}_{t+1} + \gamma \hat{T}^{out}_{t+1} + e_{t+1} \tag{2.30}$$

$$e_{t+1} = \sum_{j=0}^{L-1} \frac{\hat{T}_{t-j} - T_{t-j}}{L} \tag{2.31}$$

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. In brief, we utilize the abstract physical model for generating sampled states sequences in both the model-assisted SSL stage and model-assisted RU stage, and we will explain it in the following paragraphs.

Specifically, 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 $\{\mathbf{x}|\mathbf{x} \subset D\}$, and we denote the batch size of $\mathbf{x}$ 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 $d \in \mathbf{x} \subset D$ 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 $\{(d_i, y_i)|d_i$ equals to d except that the indoor temperature in d at time step i is set to -1, y_i is the value of indoor temperature in d at time step i, $d \in \mathbf{x} \subset D\}$. 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

$$\Theta_i = \Phi_i - \eta_2 \nabla_{\Phi_i} \mathcal{L}_{MSE}(\Phi_i), \tag{2.32}$$

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 $\Theta_i^{'}$, as described in the following equation

$$\Theta_i^{'} = \Theta_i - \eta_3 \nabla_{\Theta_i} \mathcal{L}_{target}(\Theta_i), \tag{2.33}$$

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 $\Theta_i^{'}$ 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 (2.32), while $\Theta_i^{'} - \Phi_i$ provides a better updating direction for the current knowledge extraction step compared to the Equation (2.32). Thus, we determine the true updating step for the initial model Φ_i as

$$\Phi_i = \Phi_i - \eta_1(\Theta_i^{'} - \Phi_i) \tag{2.34}$$

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.

2.3.4 Experimental Results

2.3.4 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 ϖ simulation steps and not always happens individually among 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 (NRMSE) between the prediction and the actual temperature value.

2.3.4 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 [65] is utilized for all neural networks' training. We use the EnergyPlus [110] 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) [67]. 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 [68]. The hyper-parameter settings mentioned in the previous sections are shown in Table 2.14.

2.3.4 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

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

Table 2.14: Hyper-parameters used in our experiments.

four-zone building under different sensor fault patterns.

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 2.15 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 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 2.16 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. 2.14 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.

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 ϖ selected from 0 to 5. The comparison between our approach and the standard DQN is presented in Table 2.17 and Table 2.18. 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 ϖ from 1 to 5).

	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 2.15: 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 2.16: 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.

	ϖ	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 2.17: Comparison between standard DQN controller and our sensor fault-tolerant framework (FTF) on a single-zone building under continuous faults. The fault lasts for ϖ steps. θ is the avg. indoor temperature violation rate (%).

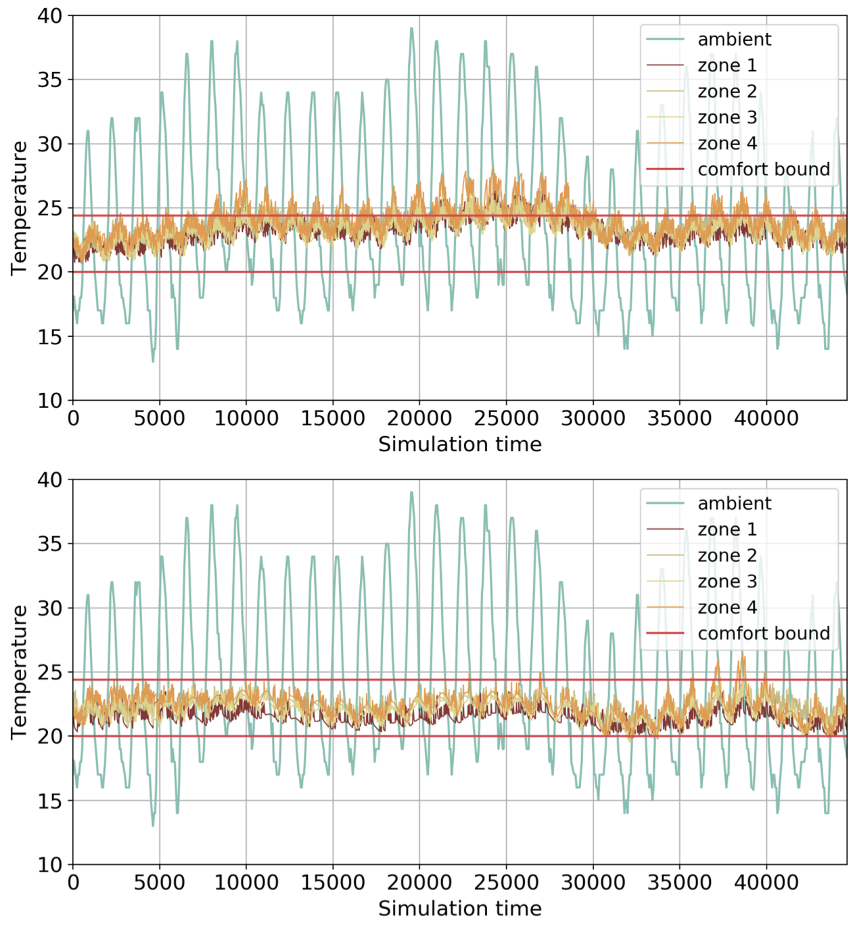


Figure 2.14: 4-zone building temperature under IID faults with $p_1 = 0.4$ without FTF (above) and with FTF control (below).

2.3.4 Evaluation of Model-Assisted Learning

In this section, we conduct experiments on the model-assisted learning algorithm and demonstrate its improvement in the performance 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 2.3.3.3 for a four-zone building.

	ϖ	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 2.18: Comparison between standard DQN controller and our sensor fault-tolerant framework (FTF) on a four-zone building under continuous faults. The fault lasts for ϖ 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 2.19: 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 [95]. 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.

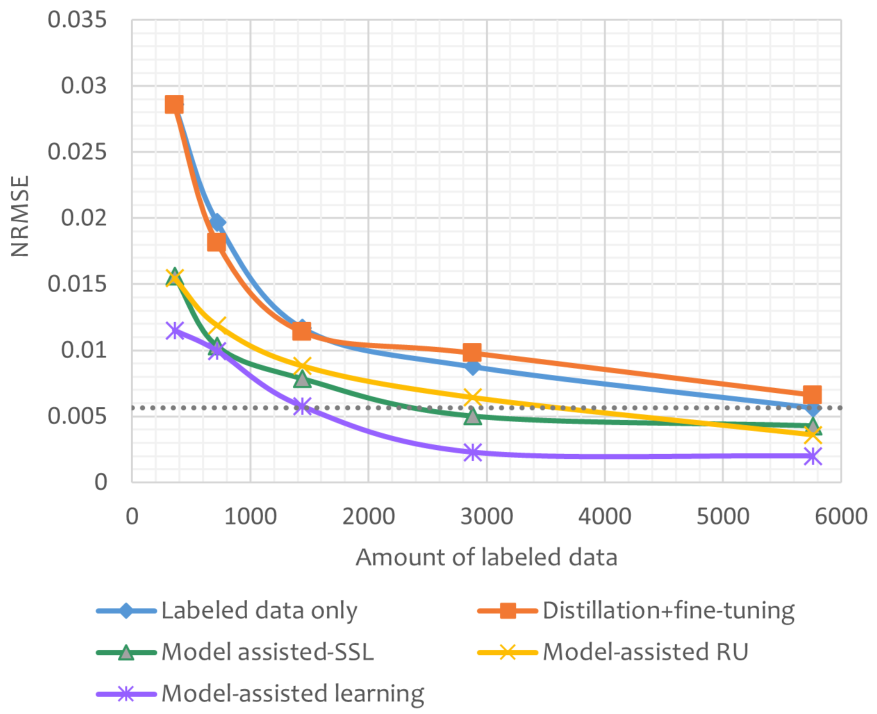


Figure 2.15: Comparison of different learning strategies on temperature prediction performance, including *Labeled data only* (blue line), *Distillation+Fine-tuning* (orange line), *Model-assisted SSL* and *RU* (green & yellow line), *Model-assisted learning* (purple line). We can observe from the figure that *Model-assisted learning* only requires around 1400 data samples to reach the Normalized RMSE 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.

The abstract model itself has the temperature prediction value with Normalized RMSE at 3.7e-2. Then if we only use the accurately labeled data collected from the building to train the neural networks in the temperature predictor module, which is shown in the first line in Table 2.19 (the model named *Labeled data only*), we can see that the Normalized RMSE remains at the relatively high level, e.g., 2.0e-2 for 1440 data samples, and 1.8e-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 [95]. 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 2.3.3.3, i.e., sampling from data space H and feeding the samples $\mathbf{x}$ to the abstract physical model M. Then we get the corresponding data pair $(\mathbf{x}, \mathbf{y})$, and train the network using $(\mathbf{x}, \mathbf{y})$ 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 2.19).

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 2.19, 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 combing 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 2.15 – it plots the same results as Table 2.19, 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.

CHAPTER 3
ADDRESSING THE DATA CHALLENGES IN VISION

In the previous chapter, we addressed the issue of data limitations in building HVAC control applications. In this chapter, we shift our focus to another important application, namely image classification. Image classification plays a crucial role in many cyber-physical systems, such as autonomous vehicle [4], agriculture [7], surgery [125], manufacturing [126], etc. To be specific, we mainly introduce two topics on the data insufficiency and data imperfection in image classification tasks,

- **Weak Adaptation Learning:** Addressing the challenge of data insufficiency in target domain, we propose a novel approach called weak adaptation learning (WAL). Our approach leverages unlabeled data from a similar source domain, limited number of labeled target domain data, and a low-cost weak annotator. Our approach includes a theoretical analysis on the error bound of the trained classifier and a multi-stage WAL method that improves the classifier accuracy by lowering such error bound.
- **Open Vocabulary Multi-Label Classification:** In the case of extreme data insufficiency, target domain data may be entirely unavailable. To address this challenge, we developed an open-vocabulary multi-label classification framework capable of making predictions on unseen classes. In our approach, we present DM-Decoder, a novel transformer decoder for facilitating the fusion of the semantics from dual-modal information source. We also design the Pyramid-Forwarding method, a new adaptation technique that enables the framework to handle high-resolution images beyond the scope of the training data while reducing the computational cost of the vision transformer.

3.1 Weak Adaptation Learning–Addressing Cross-domain Data Insufficiency with Weak Annotator

3.1.1 Background

It's common sense that having a large number of data samples with accurate labels could enable effective supervised learning methods for improving classification accuracy. But it may be difficult to collect many data samples in some problem domains or scenarios, such as for the training of autonomous vehicles during extreme weather (e.g., fog, snow, hail) and natural disasters (e.g., mudflow), or for search and rescue robots during forest fire and earthquake. One possible solution to such problem of data unavailability is using data from other similar domains to train the target domain model and then fine-tune it with limited target domain data, i.e., through domain adaptation. Taking the aforementioned cases as examples, while there may not be much data in hailing weather, we could collect data in days with heavy rain; while it may be difficult to find images during earthquakes for large parts of America, we could collect images in Japan, where earthquakes occur more often in a different environment. However, obtaining a large amount of high-quality labeled data in these source domains could still be challenging and costly.

To address the above data insufficiency challenges across domains, we consider leveraging low-cost weak annotators that can automatically generate large quantity of labeled data based on certain labeling rules/functions, task-specific heuristics, or other methods (which may be inaccurate to some degree). More specifically, our approach considers the following setting for classification problems: There is a small amount of data samples with accurate labels collected for the target domain, which is called *target domain data* or *target data* in this paper for simplicity. There is also a large amount of unlabeled data that can be acquired from a similar but different source domain

Section 3.1 is based on our work published at [36].

(i.e., there exists domain discrepancy), which is called *source (domain) data* in this paper. Finally, there is a weak annotator that can produce weak (possibly inaccurate) labels on data samples. Our objective is to learn an accurate classifier for the target domain based on the labeled target data, the initially-unlabeled source data, and the weak annotator.

The problem we are considering here is related but different from Semi-Supervised Learning (SSL) [127]–[129] and Unsupervised Domain Adaptation (UDA) [130]–[133]. In the setting of SSL, the available training data consists of two parts – one has accurate labels while the other is unlabeled, and the two parts are drawn from the same distribution in terms of training features. This is different from our problem, where there exists domain discrepancy across the source and target domains. The objective of UDA is to adapt a model to perform well in the target domain based on labeled data in the source domain and unlabeled data in the target domain. This is again very different from our problem, where the source domain data is initially unlabeled and assigned with inaccurate labels by a weak annotator, while the target domain data has labels but its quantity is small. Another related field is Positive-unlabeled Learning (PuL) [134], [135], an approach for sample selection. The training data of PuL also consists of two parts – positive and negative data, and the task is to learn a binary classifier to filter out samples that are similar to the positive data from a large amount of negative data. However, the current PuL approaches usually conduct experiments in a single data set rather than multiple domains with feature discrepancy.

To solve our target problem, we first develop a theoretical analysis on the error bound of a trained classifier with respect to the data quantity and the weak annotator performance. We then propose a Weak Adaptation Learning (WAL) method to learn an accurate classifier by lowering the error bound. The main idea of WAL is to obtain a cross-domain representation for both source domain and target domain data, and then use the labeled data to estimate the classification error/distance between the weak annotator and the ideally optimal classifier in the target domain.

Next, all the data is re-labeled based on such estimation of weak annotator classification error. Finally, the newly-relabeled data is used to learn a better classifier in the target domain.

3.1.2 Related Works

3.1.2 Weakly- and Semi-Supervised Learning

Weakly Supervised Learning is a large concept that may have multiple problem settings [120]. The problem we consider in this paper is related to the incomplete supervision setting that is often addressed by Semi-Supervised Learning (SSL) approaches. Standard SSL solves the problem of training a model with a few labeled data and a large amount of unlabeled data. Some of the widely-applied methods [127], [128], [136], [137] assign pseudo labels to unlabeled samples and then perform supervised learning. And there are works that address the noises in the labels of those samples [138]–[140]. Our target problem is related to SSL with inaccurate supervision, but is different since we consider the feature discrepancy between the (unlabeled) source data and the (labeled) target data – a case that occurs often in practice but has not been sufficiently addressed.

Positive-unlabeled Learning (PuL) is usually regarded as a sub-problem of SSL. Its goal is to learn a binary classifier to distinguish positive and negative samples from a large amount of unlabeled data and a few positive samples. Several works [134], [135] can achieve great performance on selecting samples that are similar to the positive data, and there are also works using samples selected by PuL to perform other tasks [141], [142].

3.1.2 Importance of Sample Quantity

The training of machine learning models, especially deep neural networks, often requires a large amount of data samples. However, in many practical scenarios, there is not sufficient training data to feed the learning process, degrading the model performance sharply [100], [143], [144].

Many approaches have been proposed to make up for the lack of training samples, e.g., data re-sampling [145], data augmentation [146], metric learning and meta learning [147]–[150]. And there are works [147], [151]–[153] conducting theoretical analysis on the relation between training data quantity and model performance. These analyses are usually in the form of bounding the prediction error of the models and provide valuable information on how the sample quantity of training data affects the model performance. In our work, we also perform a theoretical analysis on the error bound of the trained model, with respect to not only the data quantity but also the performance of the weak annotator.

3.1.3 Methodology

3.1.3 Theoretical Analysis

Problem Definition and Formulation

We consider the task of classification, where the goal is to predict labels for samples in the target domain. Two types of supporting data can be accessed for training the model – source domain data and target domain data. The source domain data samples are initially unlabeled and come from a joint probability distribution $\mathbb{Q}^s$. They can be labeled by a weak annotator $\mathbf{h}^w$ (which may be inaccurate) and denoted as $D_s = \{(\mathbf{x}_s, y_s)_i\}_{i=1}^{N_s}$, where N_s is the number of source data samples. The target domain data $D_t = \{(\mathbf{x}_t, y_t)_i\}_{i=1}^{N_t}$ consists of N_t samples collected from the target distribution $\mathbb{Q}^t$. Note that $\mathbb{Q}^t$ may be different from $\mathbb{Q}^S$. And we use $\mathbb{Q}_X^s$, $\mathbb{Q}_Y^s$ and $\mathbb{Q}_X^t$, $\mathbb{Q}_Y^t$ to represent the marginal distributions of the source and target domains, respectively. Moreover, as stated before, we consider the case where there is only a small amount of target domain data, i.e., $N_t \ll N_s$. Our goal is to learn an accurate classifier for the target domain. The classifier is initialized from a parameter distribution $\mathcal{H}$, which denotes the hypothesis parameter space of all possible classifiers.

In the following analysis, we will define the classification risk of a classifier and then derive its bound. According to the PAC-Bayesian framework [154], [155], the expected classification risk of a classifier drawn from a distribution $\mathcal{Q}$ that depends on the training data can be strictly bounded. Let $\mathbf{h}_\Theta$ denote a learned classifier from the training data, and its parameter Θ is drawn from $\mathcal{Q}$. We consider that the prior parameter distribution $\mathcal{H}$ over the hypothesis is independent of the training data. And given a δ with the probability $\geq 1-\delta$ over the training data set of size m, the expected error of $\mathbf{h}_\Theta$ can be bounded as follows [156]:

$$\begin{aligned} L(\mathbf{h}_\Theta) &\leq \widehat{L}(\mathbf{h}_\Theta) + \sqrt{\widehat{L}(\mathbf{h}_\Theta)\cdot\Omega} + \Omega \\ \Omega &= \frac{2\left(KL(\mathcal{Q}\|\mathcal{H}) + \ln\frac{m}{\delta}\right)}{m-1} \end{aligned} \tag{3.1}$$

Here $L(\mathbf{h}_\Theta)$ is the expected error of $\mathbf{h}$ over parameter Θ, and $\widehat{L}(\mathbf{h}_\Theta)$ is the empirical error computed from the training set ($\widehat{L}(\mathbf{h}_\Theta) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\mathbf{x}_i, y_i)$, where $\mathcal{L}$ denotes the loss of a single training sample). In Eq. (3.1), $KL(\mathcal{Q}\|\mathcal{H})$ represents the Kullback-Leibler (KL) divergence between parameter distribution $\mathcal{Q}$ and $\mathcal{H}$. For any two distributions p, q, the specific form of their KL divergence is $KL(p\|q) = -\mathbb{E}[p\cdot\ln\frac{q}{p}]$. In most cases of mini-batch training, the training loss $\widehat{L}(\mathbf{h}_\Theta)$ is much smaller than Ω, and thus we can get a further bound as follows [151]:

$$L(\mathbf{h}_\Theta) \leq \widehat{L}(\mathbf{h}_\Theta) + 4\sqrt{\frac{\left(KL(\mathcal{Q}\|\mathcal{H}) + \ln\frac{2m}{\delta}\right)}{m}} \tag{3.2}$$

Then if we denote the model parameters of $\mathbf{h}$ before the training that are drawn from $\mathcal{H}$ as $\Theta^{\mathbf{p}}$, the KL divergence can be written as $KL(\mathcal{Q}\|\mathcal{H}) = -\mathbb{E}[\Theta\cdot(\ln\Theta^{\mathbf{p}} - \ln\Theta)]$. As aforementioned, $\mathbf{h}_\Theta$ is trained with the training data set from $\mathbf{h}_{\Theta^{\mathbf{p}}}$, and we consider that the training is optimized by gradient-based method. Thus, we can formulate that $\Theta = \Theta^{\mathbf{p}} + \nabla(\widehat{L}(\mathbf{h}_{\Theta^{\mathbf{p}}}))$. Here we omit the learning rate to simplify the formula.

The PAC-Bayesian error bound is valid for any parameter distribution $\mathcal{H}$ that is independent of the training data, and any method of optimizing $\Theta^{\mathbf{p}}$ dependent on the training set [151]. Therefore, in order to simplify the problem, we instantiate the bound as setting $\mathcal{H}$ to conform to a Gaussian distribution with zero mean ($\mu_{\mathcal{H}} = 0$) and $\text{Var}_{\mathcal{H}} = \sigma^2_{\mathcal{H}}$ variance. This simplification is the same as previous PAC-Bayesian works [151], [157]. We further assume that the parameter change of the overall model during training can also be regarded as conforming to an empirical Gaussian distribution. This Gaussian distribution is independent of model parameters if we regard the parameter updates induced by gradient back-propagation as accumulated random perturbations, i.e., each training sample corresponds to a small perturbation [157]. And we denote the mean and the variance of a single training sample as follows:

$$\begin{aligned} \mu &\triangleq \mathbb{E}\left[\nabla_{\Theta^{\mathbf{P}}}\mathcal{L}(\mathbf{x}, y)\right] \\ \sigma^2 &\triangleq \mathbb{E}\left[(\nabla_{\Theta^{\mathbf{P}}}\mathcal{L}(\mathbf{x}, y) - \mu)(\nabla_{\Theta^{\mathbf{P}}}\mathcal{L}(\mathbf{x}, y) - \mu)^T\right] \end{aligned} \tag{3.3}$$

Then, the specific formula of KL divergence to any two Gaussian distributions $p \sim \mathcal{N}(\mu_1, \sigma_1^2)$, $q \sim \mathcal{N}(\mu_2, \sigma_2^2)$ is written as follows:

$$KL(p, q) = \ln\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \tag{3.4}$$

Theorem 1. *For a classifier parameter distribution $\mathcal{H} \sim \mathcal{N}(0, \sigma^2_{\mathcal{H}})$ that is independent of the training data with size m, and a posterior parameter distribution $\mathcal{Q}$ learned from the training data set, if we assume $\mathcal{Q} \sim \mathcal{N}(\mu_{\mathcal{Q}}, \sigma^2_{\mathcal{Q}})$ and consider $\Theta^{\mathbf{P}}$, Θ as drawn from $\mathcal{H}$, $\mathcal{Q}$ respectively ($\Theta = \Theta^{\mathbf{P}} + \nabla(\widehat{L}(\mathbf{h}_{\Theta^{\mathbf{P}}}))$), the KL divergence of $\mathcal{Q}$ and $\mathcal{H}$ is bounded with symbols defined in Eq. (3.3) as follows:*

$$KL(\mathcal{Q}\,\|\,\mathcal{H}) \leq \frac{\frac{\sigma^2}{m} + \mu^2}{2\sigma^2_{\mathcal{H}}} \tag{3.5}$$

The detailed proof of Theorem 1 is presented in our paper [36]. With the above risk definition, the risk of $\mathbf{h}$ with respect to the target data distribution $\mathbb{Q}^t$ is

$$R^t(\mathbf{h}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathbb{Q}^t}\mathcal{L}(\mathbf{h}(\mathbf{x}), y) = L(\mathbf{h}_{\Theta})_{\sim\mathbb{Q}^t} \tag{3.6}$$

Besides, we define the Classification Distance of two classifiers $\mathbf{h}_1$ and $\mathbf{h}_2$ under the same domain distribution $\mathbb{P}$ as

$$\mathcal{CD}_{\sim\mathbb{P}}(\mathbf{h}_1, \mathbf{h}_2) = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}}\mathcal{L}(\mathbf{h}_1(\mathbf{x}), \mathbf{h}_2(\mathbf{x})) \tag{3.7}$$

Moreover, the Discrepancy Distance of two domains is defined as in [158]: $\forall \mathbf{h_1}, \mathbf{h_2}$, the discrepancy distance between the distributions of two domains $\mathbb{P}, \mathbb{Q}$ is

$$\mathcal{DD}(\mathbb{P}, \mathbb{Q}) = \sup_{\mathbf{h_1}, \mathbf{h_2} \in \mathbb{H}} |\mathcal{CD}_{\sim\mathbb{P}}(\mathbf{h}_1, \mathbf{h}_2) - \mathcal{CD}_{\sim\mathbb{Q}}(\mathbf{h}_1, \mathbf{h}_2)| \tag{3.8}$$

For further analysis, we also define two operators in a parameter distribution $\mathcal{H}$:

- $\oplus$: $\forall \mathbf{h_1}, \mathbf{h_2} \in \mathcal{H}$, and $\forall \mathbf{x} \in \mathbb{P}$, a new classifier $\mathbf{h_3} = \mathbf{h_1} \oplus \mathbf{h_2}$ can be acquired by conducting operator $\oplus$ on $\mathbf{h_1}$ and $\mathbf{h_2}$, and $\mathbf{h_3}(\mathbf{x}) = \mathbf{h_1}(\mathbf{x}) + \mathbf{h_2}(\mathbf{x})$.
- $\ominus$: $\forall \mathbf{h_1}, \mathbf{h_2} \in \mathcal{H}$, and $\forall \mathbf{x} \in \mathbb{P}$, a new classifier $\mathbf{h_3} = \mathbf{h_1} \ominus \mathbf{h_2}$ can be acquired by conducting operator $\ominus$ on $\mathbf{h_1}$ and $\mathbf{h_2}$, and $\mathbf{h_3}(\mathbf{x}) = \mathbf{h_1}(\mathbf{x}) - \mathbf{h_2}(\mathbf{x})$.

Error Bound Analysis

Let $\mathbf{h}^{o_s}$ and $\mathbf{h}^{o_t}$ denote the ideal classifiers that perform optimally on the source data and target data, respectively:

$$\mathbf{h}^{o_s} = \arg\min_{\mathbf{h}\in\mathcal{Q}} R^s(\mathbf{h}),\ \mathbf{h}^{o_t} = \arg\min_{\mathbf{h}\in\mathcal{Q}} R^t(\mathbf{h}) \tag{3.9}$$

In our approach, we design a classifier that learns the discrepancy between the weak annotator and the ground truth (details will be introduced in Section 3.1.3), and we denote it as $\mathbf{d}$ drawn from $\mathcal{Q}$. Thus, we can get a model that is the product of conducting the aforementioned $\oplus$ operator on $\mathbf{h}$ and $\mathbf{d}$, i.e., $\mathbf{h} \oplus \mathbf{d}$. Here $\mathbf{h}$ is designed for approximating the weak labels. And for the risk of $\mathbf{h} \oplus \mathbf{d}$, we can obtain the following relation:

Theorem 2. *For all L1 (Mean Absolute Error [159]), L2 (Mean Squared Error [160]) and their* *non-negative combination loss functions (Huber Loss [161], Quantile Loss [162], etc.), the classification risk of aforementioned* $\mathbf{h} \oplus \mathbf{d}$ *can be formulated as follows:*

$$\begin{aligned} R^t(\mathbf{h} \oplus \mathbf{d}) &= \mathbb{E}_{\mathbb{Q}_X^t} \mathcal{L}(\mathbf{h} \oplus \mathbf{d}, \mathbf{h}^w \oplus \mathbf{h}^{o_t} \ominus \mathbf{h}^w) \\ &\leq \mathbb{E}_{\mathbb{Q}_X^t} \mathcal{L}(\mathbf{h}, \mathbf{h}^w) + \mathbb{E}_{\mathbb{Q}_X^t} \mathcal{L}(\mathbf{d}, \mathbf{h}^{o_t} \ominus \mathbf{h}^w) \end{aligned} \tag{3.10}$$

Please refer to our paper [36] for detailed proof of Theorem 2. Then if we consider that the training loss $\widehat{L}(\mathbf{h})$ (which equals the average loss of all training samples) is hardly influenced by the sample quantity, and it is the same for the discrepancy between two domains [163], we can split the error bound of $\mathbf{h} \oplus \mathbf{d}$ into two parts, where one part, denoted as Δ, is not influenced by the sample quantity and the other is related to the sample quantity. According to Eq. (3.2), these two parts can be written as follows (the detailed derivation of inequalities starting from Eq. (3.10) can be found in our paper [36]):

$$\begin{aligned} R^t(\mathbf{h} \oplus \mathbf{d}) &= \mathbb{E}_{\mathbb{Q}_X^t} \mathcal{L}(\mathbf{h} \oplus \mathbf{d}, \mathbf{h}^w \oplus \mathbf{h}^{o_t} \ominus \mathbf{h}^w) \\ &\leq \Delta + 4\sqrt{\frac{KL_{\mathbf{d}}}{N_t}} + 4\sqrt{\frac{KL_{\mathbf{h}}}{N_s}} \\ &\quad + 12\sqrt{\frac{\ln \frac{2N_t}{\delta}}{N_t}} + 8\sqrt{\frac{\ln \frac{2N_s}{\delta}}{N_s}} \\ \text{where} \quad \Delta &= 2\widehat{L}_t(\mathbf{h}^w) + \widehat{L}_t(\mathbf{d}) + \widehat{L}_s(\mathbf{h}) \\ &\quad + \widehat{L}_s(\mathbf{h}^w) + \mathcal{DD}(\mathbb{Q}_X^t, \mathbb{Q}_X^s) \end{aligned} \tag{3.11}$$

Here $KL_{\mathbf{d}}$ and $KL_{\mathbf{h}}$ denote KL divergences between trained $\mathbf{d}$, $\mathbf{h}$ and $\mathcal{H}$ respectively. According to Theorem 1, this KL divergence term is influenced by the training, especially impacted by the sample quantity. We will discuss the insights obtained from this error bound in the next section, and then introduce our weak adaptation learning process that is inspired by those insights.

Observation from Error Analysis

Based on the error bound derived in Eq. (3.11), we can put efforts into the following ideas in our approach to improve the classifier performance in the target domain:

- **Performance of annotator** $(2\widehat{L}_t(\mathbf{h}^w) + \widehat{L}_s(\mathbf{h}^w))$: The supervision provided by the weak annotator can guide the model to better target the given task. Ideally, we want $\mathbf{h}^w$ to produce more accurate labels for both source and target data, reducing $2\widehat{L}_t(\mathbf{h}^w)$ and $\widehat{L}_s(\mathbf{h}^w)$ simultaneously. Practically though, we may just be able to make the annotator perform better on the source domain and cannot do much with the target domain.
- **Discrepancy between domains** $(\mathcal{DD}(\mathbb{Q}^t_X, \mathbb{Q}^s_X))$: Designing loss to quantify the discrepancy between the source and target domains is well studied in Domain Adaptation. In our approach, we propose a novel inter-domain loss (called Classified-MMD) to minimize $\mathcal{DD}(\mathbb{Q}^t_X, \mathbb{Q}^s_X)$, as introduced later.
- **Quantity of source and target samples** (N_s, N_t): First, the learning of $\mathbf{d}$ needs the supervision of the ground truth, and thus we can only use the labeled target data to train $\mathbf{d}$. Then, in our method, $\mathbf{h}$ is designed to approximate the weak annotator, and therefore it may see enough that we just use the source data to train $\mathbf{h}$. However, to further reduce $KL_{\mathbf{h}}$ according to Theorem 1, we also use target samples to train $\mathbf{h}$, which increases the sample size of training data. Moreover, since the sample quantity of source data is much larger than that of target data (i.e., $N_t \ll N_s$), $\sqrt{KL_{\mathbf{d}}/N_t}$ in Eq. (3.11) dominates over $\sqrt{KL_{\mathbf{h}}/N_s}$, and in the case of $\delta \leq 2/e$, $12\sqrt{\ln \frac{2N_t}{\delta}/N_t}$ also strictly dominates over $8\sqrt{\ln \frac{2N_s}{\delta}/N_s}$. As the result, the terms influenced by a few target samples dominates the overall error risk. Therefore, directly applying $\mathbf{h} \oplus \mathbf{d}$ to the target domain will still be impacted by the insufficient samples. However, note that $\mathbf{h} \oplus \mathbf{d}$ can produce more accurate labels for the source data than the weak annotator. Therefore, we add a final step in our learning process that utilizes re-labeled source data and conducts supervised learning with such

augmentation.

3.1.3 Learning Process

In this section, we present the detailed process of our weak adaptation learning (WAL) method, which is designed based on the observations from the above error bound analysis. The overview of our WAL process is shown in Figure 3.1. The designed network consists of three parts – (Φ_0, Φ_1, Φ_2). Φ_0 can be seen as a shared feature network for both source and target data, using typical classification networks such as VGG, ResNet, etc. Φ_1 consists of three fully-connected layers that follow the output of Φ_0. And we denote the combination of Φ_0 and Φ_1 as F_1. Φ_2 consists of two fully-connected layers that follow the output of Φ_0. The combination of Φ_0 and Φ_2 is denoted as F_2. The workflow of our method is shown in Algorithm 7.

Algorithm 7 The workflow of Weak Adaptation Learning.

```
1: Initialize parameters of network components $\Phi_0, \Phi_1, \Phi_2$.
2: Obtain dataset $D$ from the source and target data with the help of weak annotator $\mathbf{h}^w$.
3: Train $F_1 = \Phi_1 \circ \Phi_0$ using $D$, with loss function following equation $\mathcal{L} = \mathcal{L}_{KL} + \alpha\mathcal{L}_{cmmd}$.
4: Fix the parameters of $\Phi_1$ and use $F_2 = \Phi_2 \circ \Phi_0$ to fit the distance of the optimal classifier for target data
   $\mathbf{h}^{o_t}$ and the weak annotator $\mathbf{h}^w$ with the target data.
5: Generate a new dataset using both source and target data. The new labels are calculated by $y^{new} =$
   $\mathbf{h}^w(\mathbf{x}) + \Phi_2(\mathbf{h}^w(\mathbf{x}), \Phi_0(\mathbf{x}))$.
6: Initialize parameters of $\Phi_0, \Phi_1, \Phi_2$.
7: Fix $\Phi_2$ and train $F_1$ using the new dataset. The loss function follows $\mathcal{L} = \mathcal{L}_{KL} + \alpha\mathcal{L}_{cmmd}$.
8: Output classifier $F_1$.
```

Stage 1: The first goal we step on is to obtain a common representation for both the source and target data, which helps us encode the inputs while mitigating the domain discrepancy in the feature representation. We gather all the unlabeled source data and the target data without their labels and use weak annotator $\mathbf{h}^w$ to assign a label for each data sample $\mathbf{x}_i$ and $y_i^w = \mathbf{h}^w(\mathbf{x}_i)$. We denote the dataset obtained in this way as $D = \{(\mathbf{x}, y^w)_i\}_{i=1}^{N_s+N_t}$. Then we fix Φ_2 and only consider the left part of the network, which is $F_1 = \Phi_1 \circ \Phi_0$. It is normally trained by supervised learning

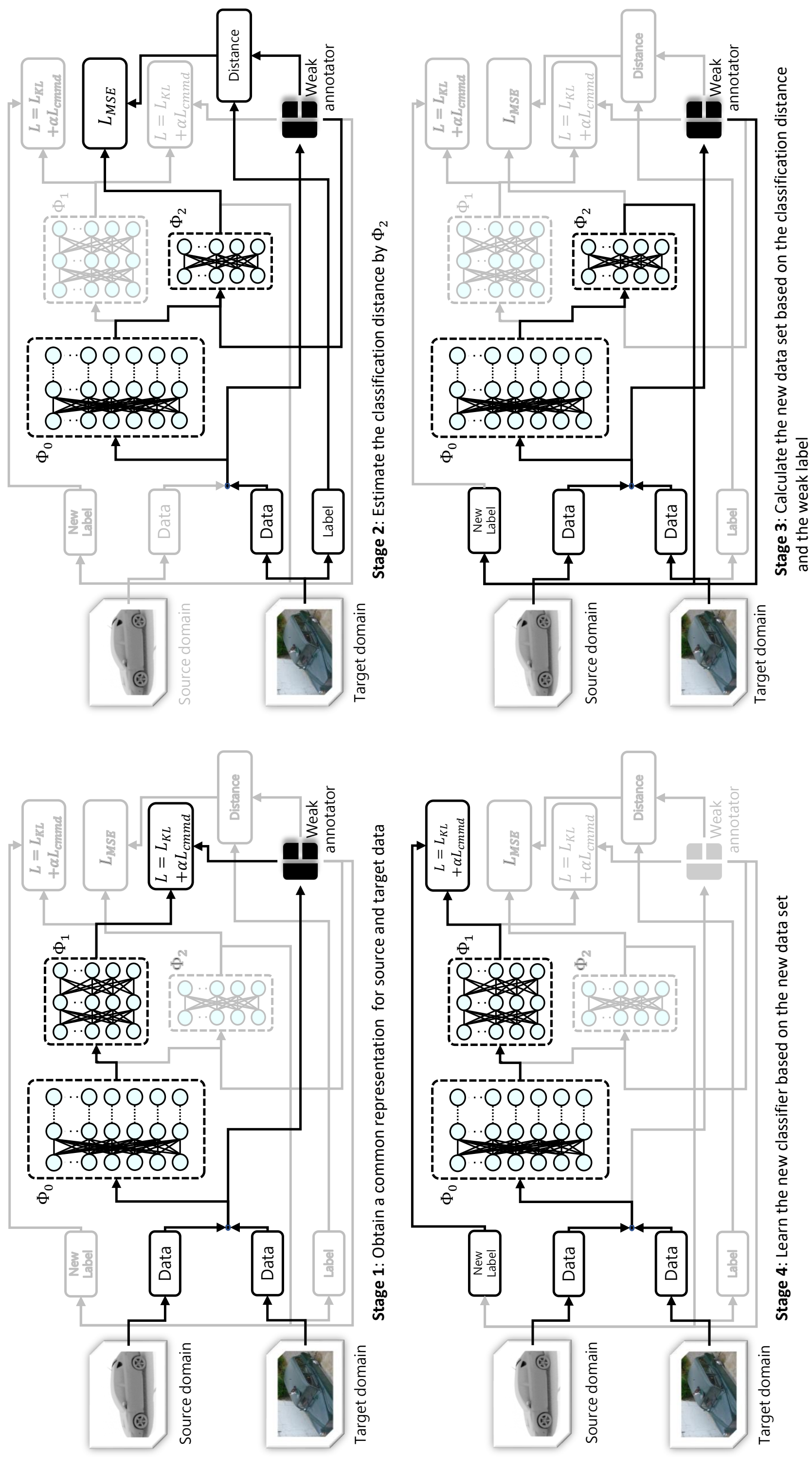

Stage 1: Obtain a common representation for source and target data
Stage 2: Estimate the classification distance by Φ_2
Stage 3: Calculate the new data set based on the classification distance and the weak label
Stage 4: Learn the new classifier based on the new data set
Source domain
Target domain
Data
Label
New Label
Φ_0
Φ_1
Φ_2
$L = L_{KL} + \alpha L_{cmmd}$
L_{MSE}
Distance
Weak annotator

Figure 3.1: Overview of the Weak Adaptation Learning (WAL) process. The designed network architecture is divided into three components Φ_0, Φ_1, Φ_2 and the algorithm has four stages. First, we use a combined loss function to learn a cross-domain representation in Φ_0 for both source and target data samples. Then, in Stage 2, Φ_2 estimates the classification distance between the weak annotator and the ideally optimal one in the target domain. A new re-labeled dataset is generated in Stage 3, and then used in Stage 4 to learn the desired classifier.

using the dataset D for ep_1 training epochs, and uses the following loss function:

$$\mathcal{L} = \mathcal{L}_{KL} + \alpha \mathcal{L}_{cmmd} \tag{3.12}$$

In this loss function, there are two loss terms and the hyper-parameter α is a scaling factor to balance the scale of two loss functions (we set it as 0.0001 in our experiments). The first term $\mathcal{L}_{KL}$ is the Kullback-Leibler (KL) divergence loss, stated as follows:

$$\begin{aligned} \mathcal{L}_{KL} &= KL(y^1_{pre} \| y^w) \\ &= KL(\Phi_1 \circ \Phi_0(\mathbf{x}) \| \mathbf{h}^w(\mathbf{x})) \end{aligned} \tag{3.13}$$

where y^1_{pre} is the output prediction value of F_1 and y^w is the corresponding weak label produced by the weak annotator $\mathbf{h}^w$. The second term $\mathcal{L}_{cmmd}$ aims to mitigate the domain discrepancy of the source and target domain at the feature representation level in the neural networks. Based on the basic MMD loss introduced by [164], we further change it into the version with data labels. We call this loss function as Classified-MMD loss (corresponding to the subscript $cmmd$), which is defined as:

$$\begin{aligned} \mathcal{L}_{cmmd} = \frac{1}{M} \cdot \sum_{i=1}^{M} \| \frac{1}{|D_X^{(S,i)}|} & \sum_{\mathbf{x}_s \in D_X^{(S,i)}} F_1(\mathbf{x}_s) \\ & - \frac{1}{|D_X^{(T,i)}|} \sum_{\mathbf{x}_t \in D_X^{(T,i)}} F_1(\mathbf{x}_t) \| \end{aligned} \tag{3.14}$$

where M is the number of classes, D_X is the data from the produced dataset D without labels, and $D_X^{(S,i)}$ is the source data selected from D_X with $\arg\max(y^w) = i$. Then, we utilize target data with its accurate labels to continue to train the network component F_1 under the loss function $\mathcal{L}_{KL}$ for ep_2 training epochs, which helps further fine-tune the feature we learned through accurate labels of the target data.

Stage 2: After finishing training in Stage 1, the next step is to estimate the distance of the optimal classifier for target data $\mathbf{h}^{o_t}$ and the weak annotator $\mathbf{h}^w$. We estimate this distance through available target data with accurate labels. We adopt the parameters trained from Stage 1 and train network component $F_2 = \Phi_2 \circ \Phi_0$ using the target data D_t. For an input data sample $\mathbf{x}$, it is brought into both Φ_0 and the weak annotator as their input. And then Φ_2 takes the output feature of $\Phi_0(\mathbf{x})$ and $\mathbf{h}^w(\mathbf{x})$ as input feature (these two features are concatenated as the input feature of Φ_2). For data sample $(\mathbf{x}_t, y_t) \in$ target dataset D_t, the learning of F_2 uses the following classifier discrepancy loss function:

$$\mathcal{L}_{MSE} = \parallel \Phi_2(\mathbf{h}^w(\mathbf{x}_t), \Phi_0(\mathbf{x}_t)) - (y_t - \mathbf{h}^w(\mathbf{x}_t)) \parallel^2 \tag{3.15}$$

The network is trained for ep_3 training epochs.

Stage 3: The third step is to generate a new dataset D_{new} through the obtained network F_2 above. Specifically, we collect data $\mathbf{x}$ from both source data and target data, and we re-label these data based on the weak annotator and F_2 obtained from the previous steps:

$$\begin{aligned} D_{new} = \{(\mathbf{x}, y^{new}) | \mathbf{x} \in D_X, \\ y^{new} = \mathbf{h}^w(\mathbf{x}) + \Phi_2(\mathbf{h}^w(\mathbf{x}), \Phi_0(\mathbf{x}))\} \end{aligned} \tag{3.16}$$

Stage 4: In the last step, we focus on $F_1 = \Phi_1 \circ \Phi_0$ again. We fix the parameters of network component Φ_2 and train F_1 using the new dataset D_{new} obtained in Stage 3. To avoid introducing feature bias from the previous steps, we clean all previous network weights and re-initialize the whole network before training. The training lasts for ep_4 epochs, and the loss function for this step is $\mathcal{L} = \mathcal{L}_{KL} + \alpha\mathcal{L}_{cmmd}$, which is the same as the function in Stage 1. Finally, we get the final model F_1 as the desired classifier.

To sum up, in Stage 1, we learn the model $\mathbf{h}$ with the help of the weak annotator to decrease the empirical loss $\widehat{L}_s(\mathbf{h})$, and the CMMD loss will reduce the term $\mathcal{DD}(\mathbb{Q}_X^t, \mathbb{Q}_X^s)$. Stage 2 uses a new classifier $\mathbf{d}$ to learn the classification distance corresponding to the term $\widehat{L}_t(\mathbf{d})$. The Stage 3 uses the annotator and the learned $\mathbf{d}$ to give more accurate labels than those given solely by the annotator. Then in Stage 4, the model is trained by the relabeled data, making both $\widehat{L}_s(\mathbf{h})$ and $\mathcal{DD}(\mathbb{Q}_X^t, \mathbb{Q}_X^s)$ be further decreased.

3.1.4 Experimental Results

3.1.4 Dataset

The experiments are conducted on three application scenarios, the digits recognition with domain discrepancy (SVHN[165], MNIST[166] and USPS[167] digit datasets), object detection with domain discrepancy (VisDA-C[168]), and object detection without domain discrepancy (CIFAR-10[169]).

3.1.4 Training Setting

All experiments are conducted on a server with Ubuntu 18.04 LTS with NVIDIA TITAN RTX GPU cards. The implementation is based on the Pytorch framework. The scaling factor α in our algorithm is set to $1e-4$. The learning rate is selected from [1e-1, 1e-2, 1e-3, 1e-4, 1e-5], for the value with the best performance in experiments. The training epochs are empirically set as multiples of 10 and are selected for each experiment. We pre-run each experiment to determine the epoch value and stop training when the performance does not increase in the next 20 epochs to prevent over-fitting. We get weak annotators in different performance by applying early stop for the training.

In the digit experiments, the training epochs in each training step is chosen as: $ep_1 = 90$, $ep_2 = 90$, $ep_3 = 40$, $ep_4 = 180$. The learning rate for experiment M $\rightarrow$ S is set to $1e-4$ and for other experiments set to $1e-5$. Training batch size is set to 128. For the baseline B_t, it is trained for 90 epochs, and the learning rate is $1e-5$ ($1e-4$ for M $\rightarrow$ S). For B_{f_1}, it is trained on the source data with weak labels for 90 epochs and on the target data for 90 epochs, and the learning rate is $1e-5$ ($1e-4$ for M $\rightarrow$ S). For B_{f_2}, it is trained on the source data with weak labels for 90 epochs and on the target data for 90 epochs, and the learning rate is $1e-5$ ($1e-4$ for M $\rightarrow$ S). Moreover, image augmentation techniques (provided by Torchvision.Transform) are applied for baselines B_t, B_{f_1}, B_{t_2}, and our approach. Other baselines use their original augmentation setting. We use the function in the Pytorch vision package for the implementation, and the images may be rotated from -3 to 3 degree, or changed to gray-scale with a probability of 0.1.

In the VisDA-C experiments, the training epochs in each training step is chosen as: $ep_1 = 90$, $ep_2 = 90$, $ep_3 = 40$, $ep_4 = 180$. The learning rate for experiment is set to $1e-5$. Training batch size is set to 128. For the baseline B_t, it is trained for 90 epochs, and learning rate is $1e-5$. For B_{f_1}, it is trained on the source data with weak labels for 90 epochs and on the target data for 90 epochs, and the learning rate is $1e-5$. For B_{f_2}, it is trained on the source data with weak labels for 90 epochs and on the target data for 90 epochs, and the learning rate is $1e-5$. The image augmentation techniques are also applied for baselines B_t, B_{f_1}, B_{t_2}, and our approach. Other baselines use their original augmentation setting. We similarly use the function in the Pytorch vision package for the implementation, and the images may be rotated from -3 to 3 degree, or changed to gray-scale with a probability of 0.1, or horizontally flipped with a probability of 0.5.

In the CIFAR-10 experiments, the training epochs in each training step is chosen as: $ep_1 = 40$, $ep_2 = 30$, $ep_3 = 70$, $ep_4 = 70$. The learning rate is set to $1e-3$. Training batch size is set to 128. For the baseline B_t, it is trained for 70 epochs, and the learning rate is $1e-3$. For B_{f_1}, it is trained on the source data with weak labels for 30 epochs and on the target data for 40 epochs, and the learning rate is $1e-3$. For B_{f_2}, it is trained on the source data with weak labels for 30 epochs and on the target data for 40 epochs, and the learning rate is $1e-3$. The image augmentation techniques are still applied to baselines B_t, B_{f_1}, B_{t_2}, and our approach. We use the function in the Pytorch vision package for the implementation, and the images are horizontally flipped with a probability of 0.5.

3.1.4 Baseline Experiments Setting

We conduct comparison experiments with the following baselines. Baseline B_{wa} is the performance of the weak annotator chosen in the experiments in the target domain. Baseline B_t is training F_1 only with target data. Baseline B_{f_1} is a fine-tuning result. It takes the same model as F_1 and first uses source domain data and weak labels generated by the weak annotator to train it. Then it uses target domain data to fine-tune the last three layers. Baseline B_{f_2} is also a fine-tuning result. The difference is that instead of fine-tuning the last three layers, it trains all network parameters.

As introduced before, our problem is related to the Semi-Supervised Learning (SSL) and the Semi-Supervised Domain Adaptation (SSDA). For SSL, although we can replace the unlabeled data with samples drawn from another domain instead of the target domain, we cannot find a good way to incorporate the weak annotator into SSL methods for fair comparison with our approach. For SSDA, we were able to extend it to our setting for comparison. Specifically, we add 1,000 unlabeled target samples (plus 1,000 labeled target samples, and this setting will be changed accordingly in digits recognition to keep consistent settings) to meet the semi-supervised requirement, and we apply weak annotator to produce weak labels instead of accurate ones for source data. We compare our approach with the following SSDA baselines: FAN [170], MME [171], ENT [172], S+T [148], [173]. Note that to the best of our knowledge, there is no previous work with exactly the same problem setting as ours. The above changes aim at making the comparison as fair as possible. Another thing that is worth to mention is that most SSDA methods conduct adaptation on the ImageNet pre-trained models, which introduces a lot of irrelevant data information from the ImageNet dataset. Thus, we disable the pre-training and only allow training with the available data.

3.1.4 Results of Digits Recognition

We evaluate our methods on the digit recognition datasets: SVHN (S), MNIST (M),and USPS (U). According to the results shown in Table 3.1, when the weak annotator performs much worse than the model learned only from the provided target data B_t (B_{wa} = 73.28% on M $\rightarrow$ U, 73.28% on S $\rightarrow$ U and 76.41% on S $\rightarrow$ M), its corresponding baseline B_{f_1} is also lower than B_t, and only the second fine-tuning method B_{f_2} is better than or competitive with B_t. This indicates that the feature learned from the source domain data and with weak labels introduce data bias, and this bias can be mitigated when the parameters from the front layers are fine-tuned by the target data.

Overall, we can clearly see that with 15,000 source domain data, limited number of labeled target domain data (second line), and a weak annotator, our method can outperform all the baselines in Table 3.1 with 80.00% on M $\rightarrow$ S, 95.99% on M $\rightarrow$ U, 96.36% on S $\rightarrow$ U and 97.24% on S $\rightarrow$ M.

Method	M $\rightarrow$ S(%)	M $\rightarrow$ U(%)	S $\rightarrow$ U(%)	S $\rightarrow$ M(%)
#samples	1000	300	300	1000
B_{wa}	59.06	73.28	73.28	76.41
B_t	61.14	89.20	89.27	94.79
B_{f_1}	55.68	84.58	77.24	80.41
B_{f_2}	77.92	94.10	94.92	95.52
S+T	65.70	93.67	91.21	96.21
ENT	67.89	92.62	92.02	96.42
MME	65.92	93.07	91.32	95.64
FAN	68.48	93.78	92.38	96.51
Ours	**80.00**	**95.99**	**96.36**	**97.24**

Table 3.1: The accuracy of different methods on digit datasets.

3.1.4 Results of Object Recognition

The results of various methods on the VisDA-C dataset are presented in Figure 3.2. In this task, we utilize the synthetic images as the source domain dataset, and the real-world images as the target domain dataset. And we can see from the table that the performance of the network trained only with the target data is merely 32.86%. Then, when the weak annotator is provided, it can help two fine-tune baselines B_{f_1} and B_{f_2} reach 27.67% and 35.03% respectively. As for the SSDA baselines, all of them perform very badly, and they are provided with more target samples with no labels. The best SSDA methods FAN can only achieve 32.99%. Our method can provide a result of 40.83%, which again exceeds all baselines above.

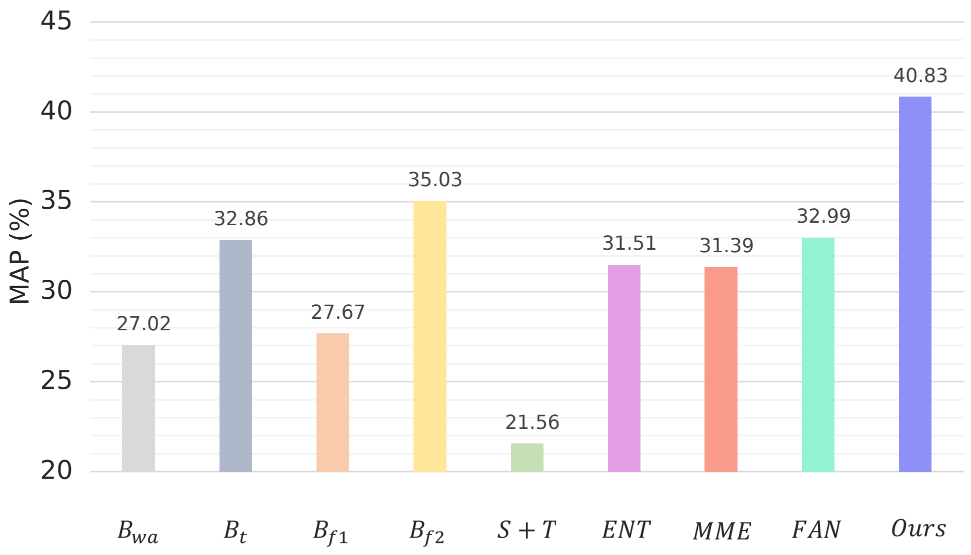


Figure 3.2: The accuracy of different methods on the VisDA-C dataset. The number is measured in percentage.

Moreover, we also test on the scenario without domain discrepancy using the CIFAR-10 dataset. We randomly select 10,000 data samples from the dataset as the source data and another 1,000 samples as the target data. The result is included in Table 3.2. As we can see, when the weak annotator is given at 48.96% accuracy, the model trained only with the target data can reach 30.46%, while our method nearly doubles the performance and hits 61.71%, which exceeds all other baselines.

Method	plane	mobile	bird	cat	deer	dog	frog	horse	ship	truck	mAP(%)
B_{wa}	43.18	65.68	28.13	25.93	29.00	**46.15**	**83.91**	41.76	72.12	51.06	48.96
B_t	19.08	63.39	03.03	30.16	25.77	22.60	46.11	50.85	23.61	25.74	30.46
B_{f_1}	57.38	77.53	38.46	33.51	45.27	33.17	73.33	58.29	57.67	60.20	52.97
B_{f_2}	44.19	80.00	38.02	41.97	47.15	24.30	78.14	55.06	**89.35**	38.19	53.49
Ours	**65.52**	**82.61**	**39.79**	**48.45**	**57.36**	43.60	67.39	**65.32**	70.42	**78.89**	**61.71**

Table 3.2: The accuracy of different methods on the CIFAR-10 dataset with 10 classes (without domain discrepancy). The number is measured in percentage. The accuracy of each class is from column 2 to column 11. The mean precision is shown in the last column.

3.1.4 Ablation Study

We also study how the quantity of target domain samples and the performance of the weak annotator affect the overall performance of our method. To reduce the impact of domain discrepancy when we study these two factors, we conduct the ablation study on CIFAR-10.

Sample Quantity of Target Samples As presented in Figure 3.3, the horizontal axis indicates the number of target domain data, and the vertical axis shows the performance of our model using the corresponding number of target domain samples. When keeping the weak annotator the same as Section 3.1.4.5 and fixing the sample quantity of the source data as 10,000, the precision of the model grows as the number of target domain data increases. And it will gradually **get saturated** when there is enough target domain data. This saturation phenomenon can be explained as the second derivative of $\sqrt{KL/N}$ and $\sqrt{\ln\frac{2N}{\delta}/N}$ for N is positive while the first derivative is negative. And according to the curve, we can observe that the performance improvement when the target data is less than the source data is relatively higher than the case when there is more target data. The reason for this can be found in our theoretical analysis, i.e., when the sample quantities of source and target data become closer, terms impacted by the quantity of target data will not dominate over the error bound.

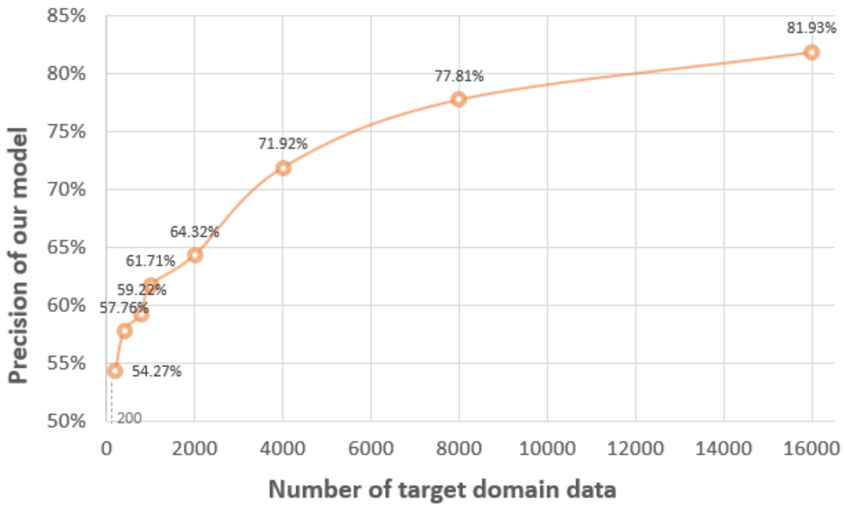


Figure 3.3: The performance of our learned model under different quantities of target domain samples.

Performance of Weak Annotator Figure 3.4 shows the curve of how the performance of our model changes with respect to the precision of the weak annotator. As shown in the figure, when the weak annotator performs the worst with accuracy of 23.79%, our model can reach 42.29%, which is a relatively significant improvement. And as the precision of the weak annotator increases, our model performs better accordingly. Interestingly, the improvement curve in Figure 3.4 is approximately linear, which demonstrates that it is reasonable to **linearly** add the terms of the weak annotator in the error bound.

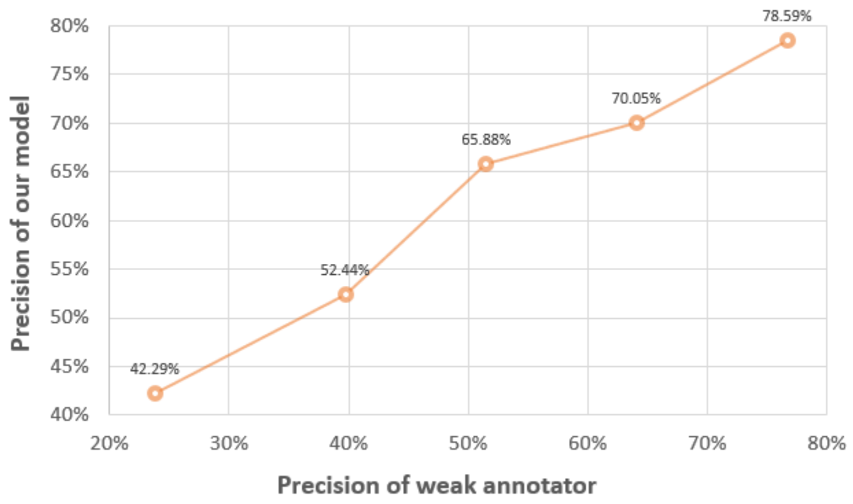


Figure 3.4: The performance of our learned model under different accuracy of the weak annotator.

3.2 Open Vocabulary Multi-Label Classification with Dual-Modal Decoder on Aligned Visual-Textual Features

3.2.1 Background

In previous section, we consider the scenario when the data in target domain is limited, then we incorporate the target domain data with unlabeled source data and as a weak annotator. But in some applications, the user need to make prediction on a wide range of classes, and the target class set might be not available in the training dataset in an extreme case.

Previously, this setting usually refers to as multi-label zero-shot learning where unseen labels may occur during testing. The related methods are often created based on label correlations [174], [175], which try to identify potential relations among the labels within the image to facilitate classification, usually, through building a label graph. Moreover, some methods [176], [177], including the previous state-of-the-art (SOTA) method ML-Decoder [178], create label embeddings solely from the words or assign learnable embeddings for each label instead of indirectly creating and

Section 3.2 is based on our work at [34].

learning from the graph relation, allowing for more sophisticated outcomes. One of the mostly used label embeddings is Word2Vec [179], which is created from pretraining tasks with external textual datasets (target label classes can be seen during pre-training). Then by extracting local discriminative features according to different label embeddings, the model outputs the per-class probability. However, these models only focus on the connection between the words, and the generalization of the learned mapping (from image to seen label) to the target mapping (from image to unseen label) is still challenging. The leveraging of the word embedding also blocks the model from handling phase/sentence labels, which also exist in practice and are very challenging to address.

In [180], the *Open-Vocabulary* setting is introduced, which is a generalization of zero-shot and weakly supervised settings and is more suitable for dealing with unseen classes. While the target classes are not known during training, it can be any subset of the entire language vocabulary in the pretraining tasks (e.g., contrastive learning on image-caption dataset). It is proved to be quite effective in some computer vision tasks, such as object detection [181] and object segmentation [182]. In this setting, instead of using costly annotations for classification datasets, the Vision-Language Pretraining (VLP) model trained on image-caption datasets can be utilized to help build the connection between the visual and textual embeddings and provide more flexibility in algorithm design. Thus instead of classic zero-shot setting, we adopt the open-vocabulary setting for our method.

On the other side, current VLP models are not silver bullets and present new challenges. Practical VLP models are usually trained with fixed low-resolution images for reducing the computational cost of large-scale data sources. The input resolution for the model adapted from those VLP models will be restricted. Besides, identifying whether a label exists in the image from their embeddings is also challenging, as the measurement based on simply comparing the cosine similarity can lead to an uncertain threshold (which is typically different for different images and objects.)

Thus, in this section, we develop a novel approach for open-vocabulary multi-label classification that significantly outperforms previous methods and provides the new state-of-the-art (SOTA) performance. In three classification tasks – open-vocabulary multi-label, single-to-multi label, and conventional multi-label – our approach provides significantly higher mean Average Precision (mAP) than prevoius methods. More specifically, to overcome the challenges in previous methods, we propose an open-vocabulary multi-label classification framework **ADDS** (**A**ligned **D**ual mo**D**ality Cla**S**sifier) based on the aligned visual and textual embeddings. The framework includes a novel **DM-decoder** (Dual-Modal decoder) design, which leverages the dual modality to enhance transformer decoder layers by progressively fusing visual embeddings with textual information and developing richer semantic understanding. It also includes a **Pyramid-Forwarding** method to adapt the model pre-trained on lower image resolutions to higher resolution images without re-training.

3.2.2 Related Works

3.2.2 Conventional Multi-label Classification

Conventional multi-label classification, which aims to classify multiple objects, scenes, or concepts in a given image, is generally a more challenging task than the typical single-label classification. It has been studied in the literature by various approaches. The first group of methods is based on the region of interest. And in previous works such as [183]–[186], multi-label classification is solved by locating each object in the image or capturing the attention map and then performing single-label classification on it. However, these methods often suffer from issues like coarse discovered regions, heavy computation costs, some concepts or scenes being hard to localize, and some regions containing duplicate concepts. Another group of methods is based on label correlations. They try to identify the potential relations among the labels within the training images to facilitate classification [187]–[189]. For instance, the method in [187] splits the feature representation into category semantics-specific representations and applies a graph neural network to explore the interactions among them. Some previous multi-label zero-shot learning methods also share a similar idea with conventional multi-label classification, which will be discussed in the next section.

3.2.2 Multi-Label Zero-Shot Classification

Some of the methods for conventional multi-label classification claim that they can also address zero-shot classification, such as [178], [183]. There are also other previous works [190]–[195]. Generally speaking, many papers in recent years try to capture the unseen labels by exploring the connections between the labels. For instance, Akata et al. [191] consider each class as an embedding in the space of attribute vectors, and then introduce a function measuring the compatibility between an image and a label embedding to determine the correct classes. The work in [192] studies the image-word relevance by estimating the principal direction for an image, which is based on the assumption that the word vectors of relevant labels for a given image can rank ahead of the irrelevant labels along a principal direction in the word vector space. The most recent paper in multi-label zero-shot learning is [178], which employs Word2Vec to generate the label text embedding and solely relies on the relation between the text features for learning. However, due to a lack of supervision on the visual information during the textual embedding learning, the learned mapping between the image and text is hard to be generalized to unseen data space.

3.2.2 Vision-Language Pre-training (VLP)

The vision language pre-training learns the semantic correspondence between image and language by pretraining on large-scale data with different tasks. In the literature, some works such as VisualBERT [196], Unicoder-VL [197], and ViLT [198] extract image tokens from the interest regions, combine them with language tokens together as the inputs, and fuse the information by the transformer in the early stage. Other works such as Contrastive LanguageImage Pre-training (CLIP) [17], Self-supervision meets Language-Image Pre-training (SLIP) [199], Bootstrapping Language-Image Pre-training (BLIP) [200], and Triple Contrastive Learning (TCL) [201] first extract the deep features of the image and the text individually, and then conduct modality interaction after the feature extraction. In this paper, we mainly maintain the alignment of the visual and textual embeddings through CLIP [17], which is built on the cosine similarity between the image and text embedding pairs and trained with a large and noisy dataset. Moreover, later in Section 3.2.4.4, we also introduce experiments on using other VLP models such as BLIP [200] and SLIP [199].

3.2.2 Open-Vocabulary Learning

VLP models enable a strong connection between images and corresponding textual information by learning from large-scale training corpora. Incorporating VLP models into model design has made arbitrary text label prediction possible, and numerous related applications have benefited from it. In recent years, open-vocabulary object detection [180], [202]–[205] and open-vocabulary semantic segmentation [182], [206], [207] have become increasingly popular. Zareian et al. [180] firstly propose Open-Vocabulary object Detection (OVD) and connect it with image-text pretraining. Gao et al. [202] leverage the localization ability of the VLP model to generate pseudo bounding-box labels for training the open-vocabulary object detector. Besides, Gu et al. [203] further distill the knowledge from a VLP model into a two-stage open-vocabulary object detector. When VLP models are leveraged for visual-semantic alignments on pixel-level information, Ma et al. [182] are able to make the zero-shot transfer to segment novel categories. In this paper, we utilize the CLIP model to keep the visual-semantic alignments to achieve open vocabulary multi-label classification.

3.2.3 Methodology

3.2.3 Overview

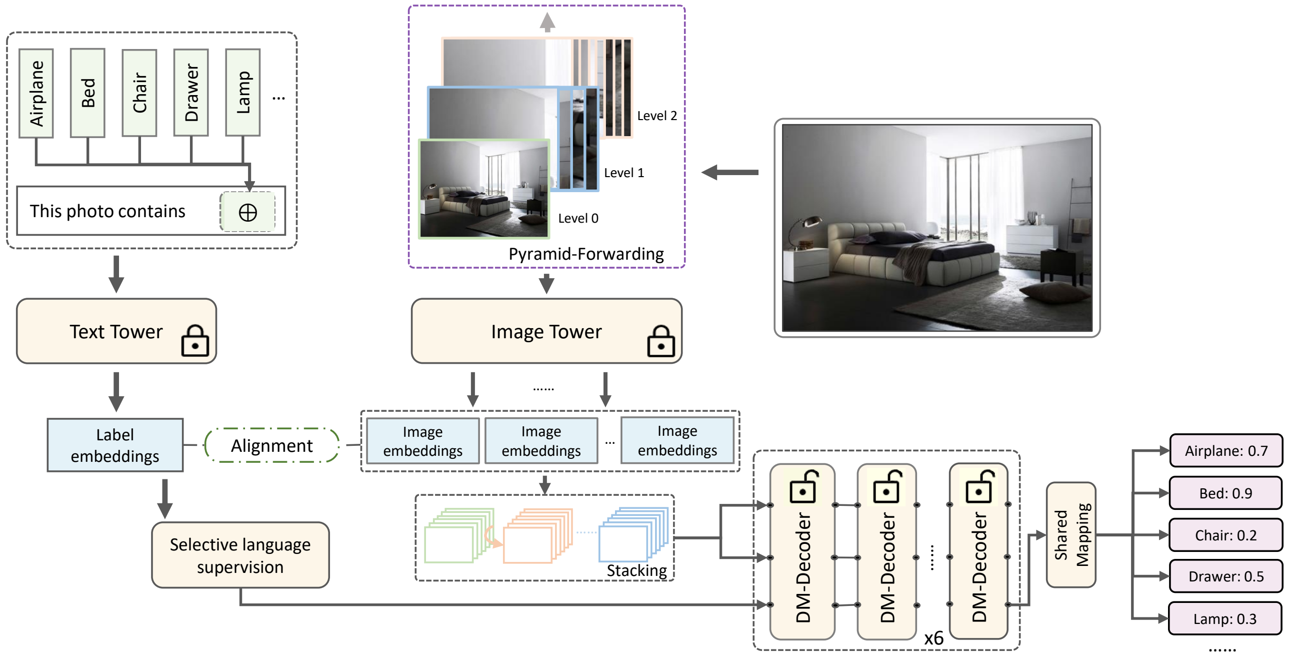


Figure 3.5: Overview of our ADDS framework for multi-label classification. Text labels with prompts are fed into a text tower to get the textual embedding. Images are first processed by a Pyramid-Forwarding module and then fed into an image tower to get the visual embeddings, which are aligned with the textual embedding and stacked on the token size dimension. Then the textual embedding (after a selective language supervision module) and the stacked visual embeddings are fused by six layers of DM-decoders with the initial query from textual embedding and the initial key/value from visual embeddings. After a shared mapping among all labels, the network outputs the probability for each label class.

In this section, we present the details of our ADDS method for multi-label classification. As shown in Figure 3.5, our method receives both the image $x_{img} \in \mathbb{R}^{H\times W\times 3}$ and the class names $X_{lbl} \subset \{\text{natural language words}\}$ as the inputs, where X_{lbl} contain words of potential labels for identifying the objects (tree, apple, computer, …), scenes (sea, sky, underground, …) or concepts (small, red, …). Then the classes names X_{lbl} are combined with prompts such as “This photo contains @" and "This is a @ photo", where $@ \in X_{lbl}$, and fed through a text tower to get the textual (class) embedding. The image input is fed through the Pyramid-Forwarding module, whose output images are then fed through an image tower to get the visual (image) embedding. The visual embedding is aligned with textual embedding, and then stacked and forwarded to the DM-decoder, whose outputs are mapped to per-class probabilities via a shared fully-connected layer. That is, given the input $\{x_{img}, X_{lbl}\}$, our model outputs $p_{pred} = [p_1, p_2, \ldots, p_k]$, where $p_i \in [0, 1]$, $k = |X_{lbl}|$.

Compared with previous works [177], [178] where the label embedding is learned from limited observations, or based on Word2Vec [178] or even randomly initialized matrix [178], a major difference of our approach is that our model is constructed based on the alignment between visual and textual embeddings. This is not only helpful for conventional multi-label classification, but also critical for boosting the performance of open-vocabulary multi-label classification. Specifically, in our setting, the training data contains the images $\{x_{(img,seen)}\}$ and the labels $X_{(lbl,seen)}$. The objective is to learn a classifier g to make predictions on an unseen image $x_{(img,unseen)}$ with unseen categories $X_{(lbl,unseen)}$, i.e., $g(x_{(img,unseen)}, x_{(lbl,unseen)}) \in \{0, 1\}$, $g() = 1$ if $x_{(img,unseen)}$ contains object/scene/concept $x_{(lbl,unseen)}$, and $g() = 0$ otherwise. Inspired by VLP, we build the visual-semantic alignment with the help of the pre-trained model from CLIP. Specifically, we employ the vision transformer (ViT) [12] network architecture as the image encoder f_{img} and the multi-layer transformer as the text encoder f_{lbl}, with the parameters of both encoders from CLIP. They are all frozen during training to maintain the alignment (may lead to worse results if unfreeze). Next, we will introduce the major components in our ADDS method in details.

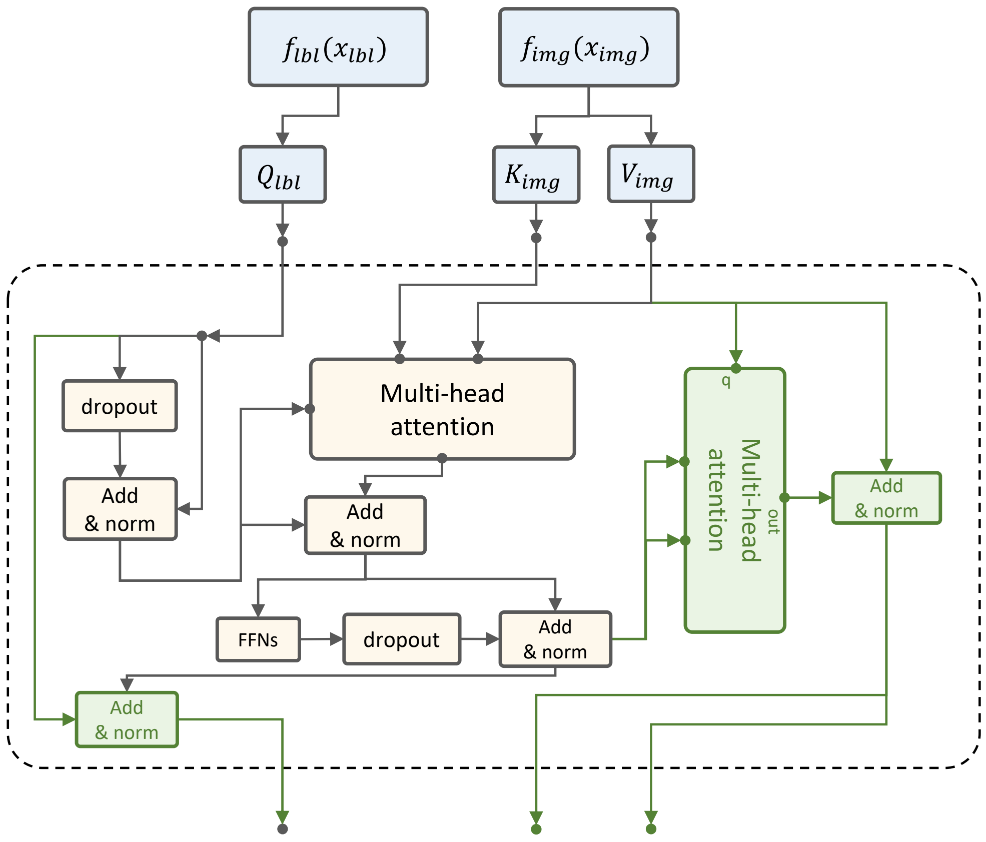


Figure 3.6: Overview of our Dual-Modal Decoder design.

3.2.3 Dual-Modal Decoder

The typical practice of aligned visual and textual embeddings in multi-label classification is to measure their cosine similarity. Given the visual embedding E_{img} from an image and a textual embedding E_{lbl} from a specific label, whether this image contains the label is determined by the calculated similarity score $s = \cos(\frac{E_{img}}{|E_{img}|}, \frac{E_{lbl}}{|E_{lbl}|})$ and a threshold η. Usually, the image is deemed to contain the label if $s > \eta$.

In practice, η can be different when considering different images and labels, which presents a significant challenge. Inspired by ML-Decoder [178], this challenge can be mitigated by turning into a binary classification task with adding decoder layers. After the visual and textual feature extraction steps, a single layer cross-attention (we omit the description of the fully-connected layer, dropout, and layer normalization for space reason) is the choice for querying the textual embedding from the visual embedding to determine the per-class probability. However, we observe some issues when we stack decoder layers similarly as in [178]:

- The model performance will often decrease after stacking more than 3 decoder layers.
- The key and value inputs are always the same from the visual embedding. As the output of the cross-attention layer is a weighted sum of its value input, the outputs of decoder layers in different levels are actually in the same (or close) semantic level and all from the same visual embedding.

To address these issues, we redesign the decoder module, with two major differences from the previous method in [178], as shown in Figure 3.6 and Equation (3.17). Specifically, we add an additional multi-head cross-attention layer MultiHdAttn_2 and use the visual embedding V_{img} to query the output Q^5_{mid} from the previous cross-attention layer MultiHdAttn_1, instead of using the textual embedding as the query (MultiHdAttn_1 utilizes the textual embedding to query the visual features). The output Q^5_{mid} contains the weighted sum of image tokens' embedding guided by the textual information, so we can redistribute them back to each image tokens' embedding through MultiHdAttn_2, to further enhance the visual embedding according to the correlation of Q^5_{mid} with the key inputs. Then after adding and normalization, the key and value input for the next decoder layer are refined by the textual information. Moreover, apart from the original skipping structures, we add an additional skipping connection from the query input to the query output, which is transformed by addition and normalization.

Formally, we denote the input query, key, value as $Q_{lbl}, K_{img}, V_{img}$, and the block's output query, key and value for the next block as $Q^{'}_{lbl}, K^{'}_{img}, V^{'}_{img}$. We output $Q^{'}_{lbl}$ only if it is the last layer. We denote DP as the dropout layer, and $\text{FFN}_1, \text{FFN}_2$ as the fully-connected layer. Then, each new decoder block can be formulated as:

$$\begin{aligned}
Q^1_{mid} &= \text{LayerNorm}(Q_{lbl} + \text{DP}(Q_{lbl})), \\
Q^2_{mid} &= \text{MultiHdAttn}_1(Q^1_{mid}, K_{img}, V_{img}), \\
Q^3_{mid} &= \text{LayerNorm}(Q^2_{mid} + Q^1_{mid}), \\
Q^4_{mid} &= \text{DP}(\text{FFN}_1(\text{ReLU}(\text{FFN}_2(Q^3_{mid})))), \\
Q^5_{mid} &= \text{LayerNorm}(Q^4_{mid} + Q^3_{mid}), \\
Q^{'}_{lbl} &= \text{LayerNorm}(Q^5_{mid} + Q_{lbl}), \\
V^1_{img} &= \text{MultiHdAttn}_2(V_{img}, Q^5_{mid}, Q^5_{mid}), \\
V^{'}_{img} &= \text{LayerNorm}(V^1_{img} + V_{img}), \\
K^{'}_{img} &= V^{'}_{img}.
\end{aligned} \tag{3.17}$$

3.2.3 Pyramid-Forwarding

A major challenge for using pre-trained models learned from large-scale datasets (e.g., the VLP models) is that they are often trained on low-resolution images (e.g., 224x224 or 336x336) due to the limitation of computational resources (training time, memory usage, etc.). Thus the extracted features are often not compatible with higher-resolution images. We could consider downsampling the higher-resolution input images to lower resolution, but that will lose significant local features. Conducting fine-tuning on higher-resolution images might slightly reduce the incompatibility with extracted features, but we still face several challenging questions: 1) Does the model support training on arbitrary scale images? 2) How much fine-tuning data are available to maintain the

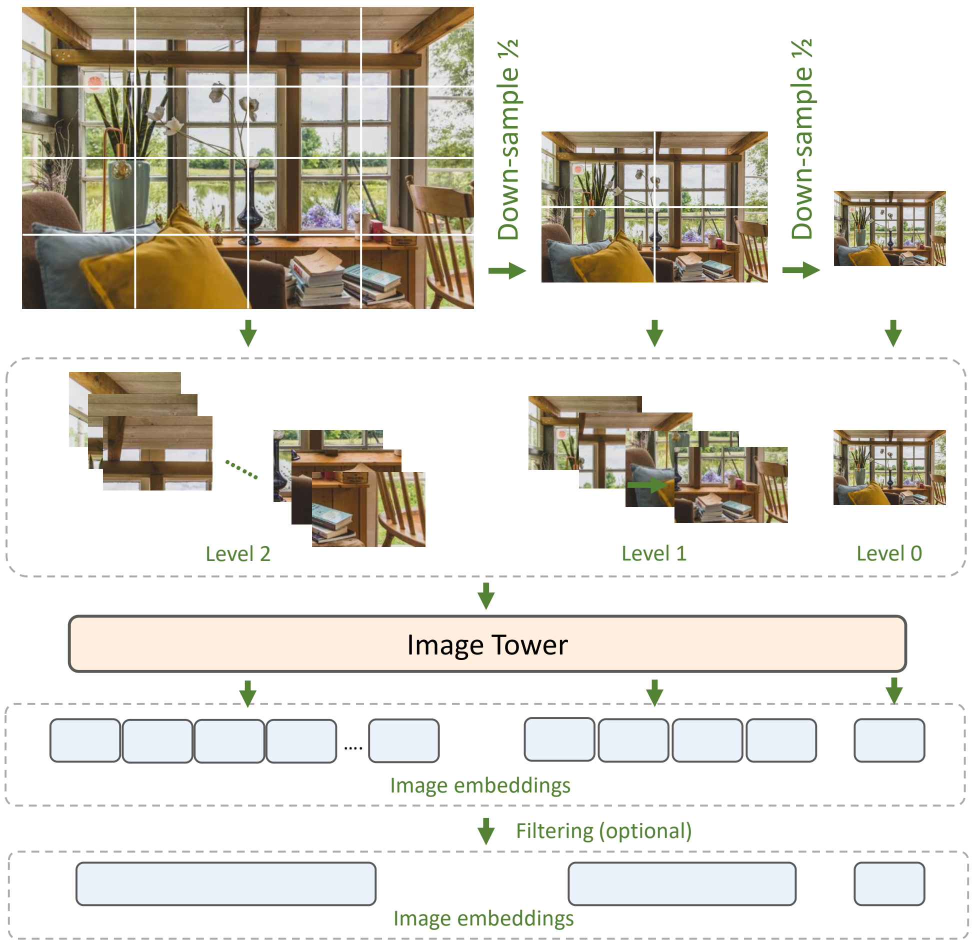


Figure 3.7: Overview of the Pyramid-Forwarding method.

model's generalization ability? 3) How long or how well the model should be trained to avoid overfitting on the fine-tuning dataset?

To address these challenges, we propose Pyramid-Forwarding, a resolution-compatible method that enables deploying a pre-trained model trained from low-resolution images on high-resolution images without retraining. This method applies to many pre-trained image decoders, and below we use the ViT [12] pre-trained on $S_{Img} \times S_{Img}$ resolution images as an example. It will be applied on $S_{Img} * d \times S_{Img} * d$ target images (d is a positive integer), and our method helps reduce the computation cost from $O(d^4 \cdot c)$ to $O(d^2 \cdot c)$, where c is a constant for all resolutions.

Specifically, given a pre-trained model with $S_{Img} \times S_{Img}$ resolution and an image of size $S_{Img} * d \times S_{Img} * d$, Pyramid-Forwarding constructs $\log(d) + 1$ levels. First we assume that $\log(d)$ is an integer, and we will discuss the non-integer case later. In level $i \in [0 \ldots \log(d)]$, the image is resized to $S_{Img} * 2^i \times S_{Img} * 2^i$ and split into $(i+1) \times (i+1)$ patches, as visualized in Figure 3.7. Then these patches are fed into the image encoder (usually wrapped into the batch dimension for parallel processing on GPUs) to obtain the feature tokens. In ViT, these tokens are stacked on the token dimension and then feed into the decoder, and the computation cost on the $S_{Img} * d \times S_{Img} * d$ image is actually $O((d^2)^2) = O(d^4)$ times more than the $S_{Img} \times S_{Img}$ image because of the self-attention layer. However, with Pyramid-Forwarding, this computation cost is reduced to $1 + 2^2 + 4^2 + \cdots + (2^{\log(d)})^2 = O(d^2)$. If $\log(d)$ is not an integer, the size of the top-level image will be changed to S^o_{Img}, and we allow the divided patches to be overlapped with their neighborhoods, while the total number of patches in this level i is still $(i+1)^2$. In addition, the computation cost of Pyramid-Forwarding can be further reduced by disposing the image patches from the non-bottom levels $i \in [1 \ldots log(d)]$, which provides a trade-off between the accuracy and the computational cost. A special case that works well is when each non-bottom level only contains the token embedding corresponding to the [CLS] token in ViT. It significantly reduces memory usage while moderately degrades the performance, when compared with the complete design.

3.2.3 Selective Language Supervision

As the label classes can be object/scene/concept names that are selected from the natural language, the total number of the labels can be extremely large (e.g., 1k in ImageNet-1k, 21k in ImageNet-21k, or even more). This presents a challenge in the training stage – if we feed all the labels into the network, it is a heavy burden on the inference speed and the memory usage. Thus we propose a *selective language supervision* method that utilizes the positive labels and part of the negative labels selected during the network training, and consider the data distribution of the positive and negative samples for reducing data imbalance.

Specifically, given multi-label $L = \{l_1, l_2, \dots, l_k\}$ from the training batch B with k classes in total, $S_{pos} = \{i | l_i = 1, l_i \in L, L \in B\}$, and $S_{neg} = \{1, 2, \dots, k\} - S_{pos}$. Then the selected label set for batch B training is

$$S^{'} = S_{pos} \cup S_{slt}, \tag{3.18}$$

where elements in S_{slt} is randomly selected from S_{neg}. $|S_{slt}| = \min(\alpha * |S_{pos}|, k - |S_{pos}|)$, where α is a hyper-parameter balancing the number of positive and negative samples (we choose $\alpha = 3$ in our experiments if not mentioned specifically). Note that the selective language supervision is only utilized in experiments with large label set (e.g., the ImageNet-21k dataset).

3.2.4 Experimental Results

3.2.4 Implementation Settings

Our code is implemented in PyTorch 1.10.2. All experiments are conducted on a server cluster running Ubuntu 18.04.6 and equipped with NVIDIA V100 GPUs. We conduct the experiments on NUS-WIDE [208], MS-COCO [209], ImageNet-1k [210], and ImageNet-21k [211] datasets. All models are optimized by the Adam optimizer [65]. The learning rate is set as 3×10^{-4} for all models trained on 224×224 and 336×336 images, and as 1×10^{-4} for all models trained on images larger than 336×336. We test the input resolution of the images on $224 \times 224, 336 \times 336, 448 \times 448, 640 \times 640, 1344 \times 1344$, and they are specified in each experiment. We adopt the ViT [12] as the image encoder and use CLIP [17] pre-trained weights. The output of ViT is $[bs, n_t, l]$, where

$bs = 56$ is the batch size, n_t is the number of output tokens from the vision transformer, and l is the embedding length determined by the type of the vision transformer. All image and text towers in our method are fixed during the training. The model is trained for 40 epochs, and the weight decay is 10^{-4}. Our model applies the loss function ASL from [212], which is also used in the most recent multi-label classification works [177], [178].

3.2.4 Open-Vocabulary Multi-label Classification

In open-vocabulary multi-label classification, we first show our results on the NUS-WIDE dataset in Table 3.3. The marking ‘uf’ means unfreezing the backbone network and ‘f’ means freezing the backbone network, and both markings are only used when ML-Decoder is switched to the same backbone as our model. To begin with, we use image-encoder ViT-B-32 and its corresponding text encoder with CLIP pre-trained weights. Using 224×224 input images, our ADDS method already achieves a 36.56% mAP, 5.46 points higher than the previous SOTA method ML-Decoder with TresNet-L as the backbone network architecture and trained on 448×448 input images. Then we use image-encoder ViT-L-336 and its corresponding text encoder with CLIP pre-trained weights on 336×336 images. The result further improves to a 39.01% mAP, 7.9 points higher than ML-Decoder (which uses higher resolution images than both of our first two settings). Finally, our result can reach 42.67% when trained on 448×448 images. We also show that when we use the same backbone network architecture ViT-L-336 and the same pretraining weights (with freezing or unfreezing the backbone) for ML-Decoder, our ADDS method still shows significant advantage (i.e., 39.01% mAP over 33.7% mAP for 336×336 images; note that ML-Decoder cannot be trained with 448×448 images for ViT-L-336 as it does not support input resolutions different from the original model resolution). Overall, the results in Table 3.3 clearly demonstrate that **our approach provides significant improvement over previous methods on open-vocabulary**

multi-label classification.

Method	Type	mAP(%)	F1 (k=3)	F1 (k=5)
CONSE	zsl	9.4	21.6	20.2
LabelEM	zsl	7.1	19.2	19.5
Fast0Tag	zsl	15.1	27.8	26.4
One Attention per Label	zsl	10.4	25.8	23.6
One Attention per Cluster (M=10)	zsl	12.9	24.6	22.9
LESA	zsl	19.4	31.6	28.7
BiAM	zsl	26.3	33.1	30.7
Generative ML-ZSL	zsl	25.7	32.8	29.3
SDL	zsl	25.9	30.5	27.8
ML-Decoder (TresNet-L, 448x448)	zsl	31.1	34.1	30.8
ML-Decoder (ViT-L-336, uf, 336x336)	zsl	16.6	16.2	17.8
ML-Decoder (ViT-L-336, f, 336x336)	zsl	33.7	31.0	32.1
ADDS (ViT-B-32, 224x224)	ov	**36.56**	**34.22**	**36.65**
ADDS (ViT-L-336, 336x336)	ov	**39.01**	**36.96**	**39.28**
ADDS (ViT-L-336, 448x448)	ov	**42.67**	**38.27**	**40.49**

Table 3.3: Comparison of different methods on open-vocabulary multi-label classification for the NUS-WIDE dataset. In the *Type* column, *zsl* means the zero-shot learning setting, *ov* means the open-vocabulary setting. Our ADDS method provides significantly better results than previous methods, including CONSE [190], LabelEM [191], Fast0Tag [192], One Attention per Label [193], One Attention per Cluster (M=10) [194], LESA [194], BiAM [195], Generative ML-ZSL [213], SDL [176], and ML-Decoder [178].

Moreover, Table 3.4 shows the experimental results on the MSCOCO dataset. We make the data splitting in the following way: after sorting the class names in increasing alphabetical order, we select the first 65 classes to be the seen classes, and the rest 15 classes to be the unseen classes. The results show that our ADDS method can achieve much better results than the original ML-Decoder (59.18% vs. 30.69%), and when using the same backbone network, it still **significantly outperforms the ML-Decoder** (54.52% vs. 43.84%; note that ML-Decoder cannot take 448×448 images with ViT-L-336 as backbone).

Method	Input Resolution	mAP(%)	F1(k=3)
ML-Decoder (ViT-L-336, f)	336x336	43.84	35.08
ML-Decoder (ViT-L-336, uf)	336x336	43.75	17.09
ML-Decoder (TresNet-L)	448x448	30.69	16.69
ADDS (ViT-L-336)	336x336	**54.52**	**51.52**
ADDS (ViT-L-336)	448x448	**59.18**	**77.34**

Table 3.4: Comparison on MSCOCO dataset for open-vocabulary multi-label classification.

3.2.4 Single-to-multi Label Classification

With overlapped classes				
Model	Backbone	mAP(%)	F1(k=3)	F1(k=5)
ML-Decoder	TResNet-L	41.60	17.80	17.80
ADDS	ViT-B-32	**59.27**	**45.88**	**45.88**
ADDS	ViT-L-336	**67.10**	**50.86**	**50.86**
Without overlapped classes				
Model	Backbone	mAP(%)	F1(k=3)	F1(k=5)
ML-Decoder	TResNet-L	38.37	6.90	6.90
ADDS	ViT-B-32	**64.32**	**30.90**	**30.90**
ADDS	ViT-L-336	**69.60**	**33.16**	**33.16**

Table 3.5: Comparison on single-to-multi label classification. Models are trained on ImageNet-1k and tested on MS-COCO. Our method greatly outperforms ML-Decoder.

We compare our approach with the previous method ML-Decoder in an extreme case of open-vocabulary multi-label classification, where models are trained on the single-label ImageNet-1k dataset and tested on the multi-label MS-COCO and NUS-WIDE datasets. We show two cases, depending on whether the testing dataset contains the overlapped classes with ImageNet-1k or not. The results are shown in Tables 3.5 and 3.6. Our approach greatly outperforms ML-Decoder, despite our model uses lower-resolution images (224×224 for ViT-B-32 backbone and 336×336 for ViT-L-336 backbone) than ML-Decoder (448×448). This again shows that **our approach**

With overlapped classes				
Model	Backbone	mAP(%)	F1(k=3)	F1(k=5)
ML-Decoder	TResNet-L	14.15	7.07	7.30
ADDS	ViT-B-32	**27.34**	**20.39**	**20.39**
ADDS	ViT-L-336	**31.07**	**24.69**	**24.69**
Without overlapped classes				
Model	Backbone	mAP(%)	F1(k=3)	F1(k=5)
ML-Decoder	TResNet-L	13.19	6.26	6.27
ADDS	ViT-B-32	**26.96**	**16.07**	**16.07**
ADDS	ViT-L-336	**30.66**	**19.18**	**19.18**

Table 3.6: Comparisons of single-to-multi label classification task which is trained on ImageNet-1k and tested on NUS-WIDE dataset. Our method shows the SOTA performance.

greatly outperforms previous methods in single-to-multi label classification.

3.2.4 Additional Experiments

Conventional Multi-label Classification

Apart from the open-vocabulary multi-label classification, we are also curious about how our model works on conventional multi-label classification. We conduct experiments on the MS-COCO dataset, and the results are shown in Table 3.7. We run a few variations of our approach and compare them with a number of baselines. We first use the ViT-L backbone [12] to test on 224×224 resolution images. We observe that our approach achieves an mAP of 89.82%, which is 5.6 points higher than the previous SOTA method ML-Decoder [178] on 224×224 images. We then switch to a larger image encoder ViT-L-336 for image resolution of 336×336. Our approach achieves an mAP of 91.76%, which is even better than what ML-Decoder can achieve on higher-resolution images of 640×640. And when we train our model with ViT-L-336 on 640×640 images, the mAP reaches 93.41%, 2.0 points higher than ML-Decoder at the same resolution. With Pyramid-Forwarding, our model can also be deployed efficiently on an even higher resolution of

Model	Backbone	Input Resolution	mAP(%)
ML-GCN	ResNet101	448x448	83.0
KSSNET	ResNet101	448x448	83.7
SSGRL	ResNet101	576x576	83.8
MS-CMA	ResNet101	448x448	83.8
ASL	TResNet-L	448x448	88.4
Q2L	TResNet-L	448x448	89.2
ML-Decoder	TResNet-M	224x224	84.2
ML-Decoder	TResNet-L	448x448	90.1
ML-Decoder	TResNet-XL	640x640	91.4
ML-Decoder	ViT-L-336 (uf)	336x336	88.5
ML-Decoder	ViT-L-336 (f)	336x336	90.6
ADDS	ViT-L	224x224	**89.82**
ADDS	ViT-L-336	336x336	**91.76**
ADDS	ViT-L-336	640x640	**93.41**
ADDS	ViT-L-336	1344x1344	**93.54**

Table 3.7: Comparison on conventional multi-label classification for the MS-COCO dataset. Our ADDS approach shows significant improvement over previous methods, including ML-GCN [214], KSSNET [215], SSGRL [187], MS-CMA [186], ASL [212], Q2L [177], and ML-Decoder [178].

1344×1344 with the model ViT-L-336 pre-trained on 336×336 resolution, and our approach can achieve an mAP of 93.54% (with much less computational cost than pre-training a model on 1344×1344 images).

Ablation Study: Effectiveness of DM-Decoder

We validate the effectiveness of our Dual-Modal decoder (DM-Decoder) design. In particular, we conduct the open-vocabulary multi-label classification experiments on NUS-WIDE. We replace the DM-Decoder in our approach with the decoder layer in the previous SOTA ML-Decoder under various number of stacking layers. The image resolution is 336×336, and all image encoder is chosen as ViT-L-336. As shown in Table 3.8, DM-Decoder significantly outperforms the decoder design in ML-Decoder in almost all cases, showing its effectiveness.

Model	mAP(%)	F1 (k=3)	F1 (k=5)
ADDS+ML-Decoder×1	36.15	32.45	35.50
ADDS+ML-Decoder×3	36.35	31.33	34.89
ADDS+ML-Decoder×6	36.34	29.83	33.56
ADDS+DM-Decoder×1	**36.88**	**32.95**	35.48
ADDS+DM-Decoder×3	**38.68**	**34.46**	**37.50**
ADDS+DM-Decoder×6	**39.01**	**36.96**	**39.28**

Table 3.8: Comparison between DM-Decoder and ML-Decoder on NUS-WIDE for open-vocabulary multi-label classification.

Ablation Study: Full Pyramid-Forwarding vs. Single-layer Pyramid-Forwarding

We then evaluate the effectiveness of Pyramid-Forwarding by comparing the results of using only a single layer of Pyramid-Forwarding versus using full Pyramid-Forwarding on MS-COCO. We choose this dataset since it has higher resolution and more suitable for comparing on different resolutions. In Table 3.9, the second column shows the level index in Pyramid-Forwarding. The first line shows the result of using the model on 336×336 images without Pyramid-Forwarding. The second line shows the model with Pyramid-Forwarding on 1344×1344 images, but only with the level of highest resolution (third level) and cutting into 16 patches. The third line shows the model with full Pyramid-Forwarding on 1344×1344 resolution images. We can see that using full Pyramid-Forwarding cannot provides much more performance boost.

Model	Level	Image Resolution	mAP(%)
ADDS	[0]	336x336	91.76
ADDS	[2]	1344x1344	91.79
ADDS	[0,1,2]	1344x1344	93.54

Table 3.9: Full vs. single-layer Pyramid-Forwarding.

Ablation Study: Impact of the Number of Training Classes

We claim that increasing the number of available training classes can benefit the single-to-multi label classification, and we validate this through training our model (on ViT-L-336) with the selective language supervision technique on the ImageNet-21k dataset. For a fair comparison, we filter out the overlapped classes with NUS-WIDE in ImageNet-1k, and we select the first 15k classes in ImageNet-21k without the overlapped classes with NUS-WIDE. We select 100 images for each class, so that the selected dataset contains 1.3M images which is the same level as ImageNet-1k (1.3M). The result of training on ImageNet-1k is shown on the first line of Table 3.10, and the second line shows the result of ImageNet-21k. With the number of available training classes becomes 15 times than before, the model mAP increases 6.9 points.

Train Dataset	Backbone	mAP(%)	F1(k=3)	F1(k=5)
ImageNet-1k	ViT-L-336	31.02	24.98	24.98
ImageNet-21k	ViT-L-336	**37.92**	**39.82**	**40.39**

Table 3.10: Comparisons of our method trained on the ImageNet-1k vs. ImageNet-21k dataset (filter the overlapped classes with NUS-WIDE) and tested on the NUS-WIDE dataset, with 336x336 resolution and ViT-L-336 backbone.

Image/Text Encoder	Backbone	Image Resolution	mAP(%)
BLIP	ViT-L(COCO)	384x384	2.52
BLIP	ViT-L	224x224	35.15
SLIP	ViT-L	224x224	34.15

Table 3.11: Applying other VLP models for alignment to our method on NUS-WIDE open-vocabulary multi-label classification task.

Experiments with Other VLP Models

Finally, we are also curious about how other VLP models perform under our method. We consider two examples BLIP [200] and SLIP [199]. They both have the contrastive loss similar to CLIP to ensure an alignment between the visual and textual features. In the second line of Table 3.11, the BLIP model with its ViT-L image encoder on 224×224 images shows a good result of 35.15%. SLIP in the third line shows similar performance. However, when we use a BLIP model that is fine-tuned on MS-COCO without the contrastive loss to ensure the alignment, the mAP quickly goes down to 2.52%. This strongly shows that the correlation between visual and textual embedding plays an important role in providing the performance boost, instead of the image or text encoder itself.

CHAPTER 4
CONCLUSION AND FUTURE WORK

4.1 Summary of Key Findings and Significance

This dissertation delves into the challenges of learning from limited and imperfect data in two different applications for cyber-physical systems.

In the building HVAC control task, we first address the slow and expensive process of collecting data from real and simulated building environments. To circumvent the long training time of a model-free DRL agent, we propose a transfer learning-based approach that can effectively transfer a DRL-based HVAC controller trained for a source building to a controller for a target building with minimal effort and improved performance. We show that by manually filtering out the building-specific behavior as much as possible, we are able to transfer the control policy which contains building-agnostic behaviour between buildings and largely reduce the training time of the DRL controller.

Since the source building might not be available or the similarity between two buildings can be pretty large, we then explore the setting when the knowledge from domain experts in various forms (including abstract physical models, historical data, and expert rules) is available. We design a unified framework for incorporating this expert information through different expert functions. The combined expert function can be viewed as strong prior to the control policy, and the final result shows that our learning framework can effectively accelerate the online DRL training process.

In addition to addressing the challenge of insufficient data, we also tackle the issue of data corruption in building HVAC systems. Temperature sensors in buildings are vulnerable to faults and malicious attacks, leading to incorrect temperature readings. To address this issue, we propose a learning-based framework for sensor fault-tolerant HVAC control. Before the DRL controller stage, the framework attempts to mitigate faulty readings by generating temperature predictions and selecting between the sensor reading and its prediction. The experiment results show that it can significantly reduce building temperature violations under a variety of sensor fault patterns while maintaining energy efficiency.

In the image classification task, we addressed the issue of insufficient target domain data by leveraging unlabeled source data and a low-cost weak annotator as the additional information. Through theoretical analysis, we have revealed the error bound of the trained classifier and its connections with the performance of weak annotators, domain discrepancy, and the quantity of source and target samples. Then we propose Weak Adaptation Learning, which learns a more accurate classifier in the target domain by incorporating the information from unlabeled source data and a low-cost weak annotator. Our results demonstrate the effectiveness of our approach and its potential for improving image classification accuracy with limited target domain data.

In addition, we make exploration on the data limitation challenge when the target classes do not exist in the training set. We develop an open-vocabulary multi-label classification framework based on visual-semantic alignment. The experiments show that our method significantly outperforms the previous SOTA methods in open-vocabulary multi-label classification, single-to-multi-label classification, and conventional multi-label classification task.

4.2 Opportunities for Future Research

We will show some applicable directions on the topic of learning from limited and imperfect data in cyber-physical systems.

4.2.1 Efficient Learning for Model-free DRL Building HVAC Controller

One of the challenges the building HVAC control faced is still how to build a building HVAC controller with minimal cost. The model-based methods need a sufficiently accurate physical model for runtime HVAC control while model-free methods suffer from data insufficiency and need a long training time to reach a desired performance. The possible approaches are to leverage the extra information, such as historical data, expert models, to build a strong prior for the control model.

4.2.1 Decomposed Surrogate Model with Multi-Task Learning

There's a direction which considers the sustainability and interpretability by decomposing a large model into several sub-models. Considering a complex system including various components, we want to build a neural network (or other ML models) to make predictions on the next step system state (e.g., indoor temperature). This is a crucial component for many model-based controllers and some model-free methods. Note the inputs (current and previous system states, control actions) as O_{in}, and the prediction output as O_{out}. Then the final prediction can be represented or relies some indicators in the system (they can be some intermediate observations or hidden variables), let's call them $v_1, v_2, v_3, v_4, \ldots, v_k$. These variables can have their own dependence, and we mark each set of dependents with a task number. For example, in task 1, v_1 depends on O_{in}; in task 2, in task 2, v_2 depends on v_1 and part of O_{in}; in task 3, v_4 depends on v_1, v_2, v_3 and part of O_{in}; $\ldots$ These tasks are built based on the domain knowledge and each task can be captured by an independent neural network. The training process consists of multiple tasks, including every single task, a combination of some tasks that have a direct dependency, the end-to-end training on all tasks. Then the surrogate model (prediction model) is decomposed into several sub-tasks through domain knowledge, the task complexity for each component is much lower than the overall prediction task and becomes easier to be learned. Besides, the interpretability of the model prediction is enhanced, as we can observe from the generated hidden variables to figure out the inference process. Moreover, when there are changes in the building dynamics, we can validate the effectiveness of each network component and only update the neural networks that are inconsistent with the currently collected data. It ensures the sustainability of the model.

4.2.1 Large Foundation Model for Control Tasks

The current trend in solving common vision and language tasks is to build models based on existing large pre-trained models, and leverage the fine-tuning or prompting techniques to adapt them to the target domain. For instance, there are lots of human-designed networks in vision tasks, and they are usually pre-trained with large datasets for being used as the backbone model in a wide range of vision tasks. The typical works including ResNet [10], ViT [12], MoCo [15], etc. In the NLP task domain, the pre-train models like BERT [216], GPT-2 [217], GPT-3 [218], RoBERTa [219] are frequently used for text classification, question answering, sentiment analysis, etc. There are also some Vision and Language Pre-Trained Models, such as CLIP [17], ALIGN [18], BLIP [200], GPT-4 [19] for handling the cross-modality tasks. When these models are used in NLP tasks as a backbone network, their weights are usually fixed or with little changes. So in most cases, only a small set of target data is needed for making adaptation to the target domain. In the control task domain, exploration on this direction is still at the initial stage. There's only a few works, such as the recently proposed generalized pre-training framework SMART [220], for control tasks. How to build these large foundation models for control tasks and then adapting to building-related task domains is still an open question.

4.2.2 Learning from Imperfect Data for Vision Tasks

The recent advances in vision tasks (including language tasks) have led to the widespread use of large pre-trained models. The methods based on large pre-trained models usually have significantly better performance than previous methods, and they usually require much less training data to reach a desired performance. Thus, when we consider explorations on data limitations in vision tasks, we need to take this factor into account. There are several directions that we think could be useful in this domain. For instance, an applicable direction is to explore scenarios where large pre-trained models cannot be applied. This may be due to resource limitations, such as on mobile or wearable devices, or with privacy concerns, where users may not want to upload data to a service provider. In such cases, we can either choose to avoid using the large pre-trained model and then create the solution for data limitation, or explore the way of leveraging large pre-trained models under the resource constraints (e.g., combined with cloud computation, or with model compression, etc.)

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[2] A. Tewari, O. Fried, J. Thies, *et al.*, "State of the art on neural rendering," in *Computer Graphics Forum*, Wiley Online Library, vol. 39, 2020, pp. 701–727.

[3] D. Eck and J. Schmidhuber, "A first look at music composition using lstm recurrent neural networks," *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, vol. 103, p. 48, 2002.

[4] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[5] D. Guest, K. Cranmer, and D. Whiteson, "Deep learning and its application to lhc physics," *Annual Review of Nuclear and Particle Science*, vol. 68, pp. 161–181, 2018.

[6] T. Wei, Y. Wang, and Q. Zhu, "Deep reinforcement learning for building hvac control," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.

[7] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and electronics in agriculture*, vol. 147, pp. 70–90, 2018.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[11] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[13] Z. Liu, Y. Lin, Y. Cao, *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[14] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, PMLR, 2020, pp. 1597–1607.

[15] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[16] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," *arXiv preprint arXiv:2111.06377*, 2021.

[17] A. Radford, J. W. Kim, C. Hallacy, *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8748–8763.

[18] C. Jia, Y. Yang, Y. Xia, *et al.*, "Scaling up visual and vision-language representation learning with noisy text supervision," in *International Conference on Machine Learning*, PMLR, 2021, pp. 4904–4916.

[19] OpenAI, *Gpt-4 technical report*, 2023. arXiv: `2303.08774 [cs.CL]`.

[20] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[21] V. Mnih, A. P. Badia, M. Mirza, *et al.*, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, PMLR, 2016, pp. 1928–1937.

[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[24] Z. Zhang, A. Chong, Y. Pan, C. Zhang, S. Lu, and K. P. Lam, "A deep reinforcement learning approach to using whole building energy model for hvac optimal control," in *BPAC and SimBuild*, vol. 3, 2018, pp. 22–23.

[25] G. Gao, J. Li, and Y. Wen, "Energy-efficient thermal comfort control in smart buildings via deep reinforcement learning," *arXiv preprint arXiv:1901.04693*, 2019.

[26] U. DoE *et al.*, "Buildings energy data book," *Energy Efficiency & Renewable Energy Department*, vol. 286, 2011.

[27] L. Yu, W. Xie, D. Xie, *et al.*, "Deep reinforcement learning for smart home energy management," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2751–2762, 2019.

[28] L. Yu, Y. Sun, Z. Xu, *et al.*, "Multi-agent deep reinforcement learning for hvac control in commercial buildings," *IEEE Transactions on Smart Grid*, vol. 12, no. 1, pp. 407–419, 2020.

[29] P. M. Papadopoulos, V. Reppa, M. M. Polycarpou, and C. G. Panayiotou, "Distributed design of sensor fault-tolerant control for preserving comfortable indoor conditions in buildings," *IFAC-PapersOnLine*, vol. 51, no. 24, pp. 688–695, 2018.

[30] S. Xu, Y. Wang, Y. Wang, Z. O'Neill, and Q. Zhu, "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*, 2020, pp. 230–239.

[31] S. Xu, Y. Fu, Y. Wang, *et al.*, "Accelerate online reinforcement learning for building hvac control with heterogeneous expert guidances," in *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2022, pp. 89–98.

[32] C. Lombard and E. Mathews, "Efficient, steady state solution of a time variable rc network, for building thermal analysis," *Building and Environment*, vol. 27, no. 3, pp. 279–287, 1992.

[33] K. Yun, R. Luck, P. J. Mago, and H. Cho, “Building hourly thermal load prediction using an indexed arx model,” in *Energy and Buildings*, 2012.

[34] S. Xu, Y. Li, J. Hsiao, C. Ho, and Z. Qi, “A dual modality approach for (zero-shot) multi-label classification,” *arXiv preprint arXiv:2208.09562*, 2022.

[35] S. Xu, Y. Fu, Y. Wang, Z. O’Neill, and Q. Zhu, “Learning-based framework for sensor fault-tolerant building hvac control with model-assisted learning,” in *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2021, pp. 1–10.

[36] S. Xu, L. Wang, Y. Wang, and Q. Zhu, “Weak adaptation learning: Addressing cross-domain data insufficiency with weak annotator,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8917–8926.

[37] N. E. Klepeis, W. C. Nelson, W. R. Ott, *et al.*, “The national human activity pattern survey (nhaps): A resource for assessing exposure to environmental pollutants,” *Journal of Exposure Science & Environmental Epidemiology*, vol. 11, no. 3, pp. 231–252, 2001.

[38] B. Huang, Y. Zhu, Y. Gao, *et al.*, “The analysis of isolation measures for epidemic control of covid-19,” *Applied Intelligence*, vol. 51, no. 5, pp. 3074–3085, 2021.

[39] S. Salakij, N. Yu, S. Paolucci, and P. Antsaklis, “Model-based predictive control for building energy management. i: Energy modeling and optimal control,” *Energy and Buildings*, vol. 133, pp. 345–358, 2016.

[40] M. Maasoumy, A. Pinto, and A. Sangiovanni-Vincentelli, “Model-based hierarchical optimal control design for hvac systems,” in *Dynamic Systems and Control Conference*, vol. 54754, 2011, pp. 271–278.

[41] T. Wei, Q. Zhu, and N. Yu, “Proactive demand participation of smart buildings in smart grid,” *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1392–1406, 2015.

[42] Y. Yang, S. Srinivasan, G. Hu, and C. J. Spanos, “Distributed control of multi-zone hvac systems considering indoor air quality,” *arXiv preprint arXiv:2003.08208*, 2020.

[43] M. Maasoumy, M. Razmara, M. Shahbakhti, and A. S. Vincentelli, “Handling model uncertainty in model predictive control for energy efficient buildings,” *Energy and Buildings*, vol. 77, pp. 377–392, 2014.

[44] M. Maasoumy, M. Razmara, M. Shahbakhti, and A. S. Vincentelli, “Selecting building predictive control based on model uncertainty,” in *2014 American Control Conference*, IEEE, 2014, pp. 404–411.

[45] T. Wei, S. Ren, and Q. Zhu, “Deep reinforcement learning for joint datacenter and hvac load control in distributed mixed-use buildings,” *IEEE Transactions on Sustainable Computing*, pp. 1–1, 2019.

[46] E. Barrett and S. Linder, “Autonomous hvac control, a reinforcement learning approach,” in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2015.

[47] B. Li and L. Xia, “A multi-grid reinforcement learning method for energy conservation and comfort of hvac in buildings,” 2015, pp. 444–449.

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

[49] P. Fazenda, K. Veeramachaneni, P. Lima, and U.-M. O’Reilly, “Using reinforcement learning to optimize occupant comfort and energy usage in hvac systems,” *JAISE*, pp. 675–690, 2014.

[50] G. T. Costanzo, S. Iacovella, F. Ruelens, T. Leurs, and B. J. Claessens, “Experimental analysis of data-driven control for a building heating system,” *Sustainable Energy, Grids and Networks*, vol. 6, pp. 81–90, 2016.

[51] Z. Zhang and K. P. Lam, “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*, 2018, pp. 148–157.

[52] Y. Li, Y. Wen, D. Tao, and K. Guan, “Transforming cooling optimization for green data center via deep reinforcement learning,” *IEEE transactions on cybernetics*, vol. 50, no. 5, pp. 2002–2013, 2019.

[53] A. Naug, I. Ahmed, and G. Biswas, “Online energy management in commercial buildings using deep reinforcement learning,” in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, IEEE, 2019, pp. 249–257.

[54] G. Gao, J. Li, and Y. Wen, “Deepcomfort: Energy-efficient thermal comfort control in buildings via reinforcement learning,” *IEEE Internet of Things Journal*, 2020.

[55] D. B. Crawley, C. O. Pedersen, L. K. Lawrie, and F. C. Winkelmann, “Energyplus: Energy simulation program,” *ASHRAE*, vol. 42, 2000.

[56] Y. Ma, F. Borrelli, B. Hencey, B. Coffey, S. Bengea, and P. Haves, “Model predictive control for the operation of building cooling systems,” *IEEE Transactions on Control Systems Technology*, vol. 20, no. 3, pp. 796–803, 2012.

[57] P. Lissa, M. Schukat, and E. Barrett, “Transfer learning applied to reinforcement learning-based hvac control,” *SN Computer Science*, vol. 1, 2020.

[58] Y. Chen, Z. Tong, Y. Zheng, H. Samuelson, and L. Norford, “Transfer learning with deep neural networks for model predictive control of hvac and natural ventilation in smart buildings,” *Journal of Cleaner Production*, vol. 254, p. 119 866, 2020.

[59] Y. Zhan and M. E. Taylor, “Online transfer learning in reinforcement learning domains,” in *2015 AAAI Fall Symposium Series*, 2015.

[60] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, “Learning invariant feature spaces to transfer skills with reinforcement learning,” *ICLR*, 2017.

[61] F. L. Da Silva and A. H. R. Costa, “A survey on transfer learning for multiagent reinforcement learning systems,” *Journal of Artificial Intelligence Research*, vol. 64, pp. 645–703, 2019.

[62] I. Akkaya, M. Andrychowicz, M. Chociej, *et al.*, “Solving rubik’s cube with a robot hand,” *arXiv:1910.07113*, 2019.

[63] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[64] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[65] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[66] I. Goodfellow, Y. Bengio, and A. Courville, “6.5 back-propagation and other differentiation algorithms,” *Deep Learning*, pp. 200–220, 2016.

[67] M. Wetter, “Co-simulation of building energy and control systems with the building controls virtual test bed,” *Journal of Building Performance Simulation*, vol. 4, no. 3, pp. 185–203, 2011.

[68] S. Wilcox and W. Marion, “Users manual for tmy3 data sets,” 2008.

[69] T. Hester, M. Vecerik, O. Pietquin, *et al.*, “Deep q-learning from demonstrations,” in *AAAI*, 2018.

[70] Y. Fu, S. Xu, Q. Zhu, and Z. O’Neill, “Containerized framework for building control performance comparisons: Model predictive control vs deep reinforcement learning control,” in *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2021, pp. 276–280.

[71] P. Lissa, M. Schukat, M. Keane, and E. Barrett, “Transfer learning applied to drl-based heat pump control to leverage microgrid energy efficiency,” *Smart Energy*, vol. 3, p. 100 044, 2021.

[72] J. Schepers, R. Eyckerman, F. Elmaz, W. Casteels, S. Latré, and P. Hellinckx, “Autonomous building control using offline reinforcement learning,” in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, Springer, 2021, pp. 246–255.

[73] A. Nair, M. Dalal, A. Gupta, and S. Levine, “Accelerating online reinforcement learning with offline datasets,” *arXiv preprint arXiv:2006.09359*, 2020.

[74] X. Zhang, X. Jin, C. Tripp, D. J. Biagioni, P. Graf, and H. Jiang, “Transferable reinforcement learning for smart homes,” in *Proceedings of the 1st International Workshop on Reinforcement Learning for Energy Management in Buildings & Cities*, 2020, pp. 43–47.

[75] S. Fujimoto, D. Meger, and D. Precup, “Off-policy deep reinforcement learning without exploration,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 2052–2062.

[76] N. Jaques, A. Ghandeharioun, J. H. Shen, *et al.*, “Way off-policy batch deep reinforcement learning of implicit human preferences in dialog,” *arXiv preprint arXiv:1907.00456*, 2019.

[77] Z. Wang, A. Novikov, K. Zolna, *et al.*, “Critic regularized regression,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 7768–7778, 2020.

[78] Y. Guo, S. Feng, N. Le Roux, E. Chi, H. Lee, and M. Chen, "Batch reinforcement learning through continuation method," in *International Conference on Learning Representations*, 2020.

[79] X. Chen, Z. Zhou, Z. Wang, C. Wang, Y. Wu, and K. Ross, "Bail: Best-action imitation learning for batch deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 353–18 363, 2020.

[80] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau, "Benchmarking batch deep reinforcement learning algorithms," *arXiv preprint arXiv:1910.01708*, 2019.

[81] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," in *ICML*, PMLR, 2020.

[82] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[83] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *NeurIPS*, 2021.

[84] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[85] S. Li and O. Bastani, "Robust model predictive shielding for safe reinforcement learning with stochastic dynamics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 7166–7172.

[86] O. Bastani, S. Li, and A. Xu, "Safe reinforcement learning via statistical model predictive shielding.," in *Robotics: Science and Systems*, 2021, pp. 1–13.

[87] D. T. Phan, R. Grosu, N. Jansen, N. Paoletti, S. A. Smolka, and S. D. Stoller, "Neural simplex architecture," in *NASA Formal Methods: 12th International Symposium, NFM 2020, Moffett Field, CA, USA, May 11–15, 2020, Proceedings 12*, Springer, 2020, pp. 97–114.

[88] Y. Wang, C. Huang, Z. Wang, Z. Yang, and Q. Zhu, "Joint differentiable optimization and verification for certified reinforcement learning," *arXiv preprint arXiv:2201.12243*, 2022.

[89] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *AAAI 2016*, 2016.

[90] B. W. Olesen and G. S. Brager, "A better way to predict comfort: The new ashrae standard 55-2004," 2004.

[91] U. S. D. of Labor, *Osha technical manual (otm) section iii: Chapter 2*, 2021.

[92] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[93] C.-A. Cheng, A. Kolobov, and A. Swaminathan, "Heuristic-guided reinforcement learning," *NeurIPS*, 2021.

[94] I. Uchendu, T. Xiao, Y. Lu, *et al.*, "Jump-start reinforcement learning," *arXiv preprint arXiv:2204.02372*, 2022.

[95] G. Hinton, O. Vinyals, J. Dean, *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[96] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[97] G. Brockman, V. Cheung, L. Pettersson, *et al.*, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[98] R. Judkoff and J. Neymark, "International energy agency building energy simulation test (bestest) and diagnostic method," Feb. 1995.

[99] G. Brockman, V. Cheung, L. Pettersson, *et al.*, *Openai gym*, 2016. eprint: `arXiv:1606.01540`.

[100] S. Xu, Y. Fu, Y. Wang, Z. O'Neill, and Q. Zhu, *Learning-based framework for sensor fault-tolerant building hvac control with model-assisted learning*, 2021. arXiv: `2106.14144 [eess.SY]`.

[101] J. Qin and S. Wang, "A fault detection and diagnosis strategy of vav air-conditioning systems for improved energy and control performances," *Energy and buildings*, vol. 37, no. 10, pp. 1035–1048, 2005.

[102] H. M. Newman, *BACnet: The Global Standard for Building Automation and Control Networks*. Momentum Press, 2013.

[103] D. G. Holmberg and D. Evans, *BACnet wide area network security threat assessment*. US Department of Commerce, National Institute of Standards and Technology, 2003.

[104] Z. Ma and S. Wang, "Fault-tolerant supervisory control of building condenser cooling water systems for energy efficiency," *HVAC&R Research*, vol. 18, no. 1-2, pp. 126–146, 2012.

[105] X.-B. Yang, X.-Q. Jin, Z.-M. Du, B. Fan, and Y.-H. Zhu, "Optimum operating performance based online fault-tolerant control strategy for sensor faults in air conditioning systems," *Automation in Construction*, vol. 37, 2014.

[106] V. Gunes, S. Peter, and T. Givargis, "Improving energy efficiency and thermal comfort of smart buildings with hvac systems in the presence of sensor faults," in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, IEEE, 2015, pp. 945–950.

[107] S. Wang and Y. Chen, "Fault-tolerant control for outdoor ventilation air flow rate in buildings based on neural network," *Building and Environment*, vol. 37, no. 7, pp. 691–704, 2002.

[108] X. Jin and Z. Du, "Fault tolerant control of outdoor air and ahu supply air temperature in vav air conditioning systems using pca method," *Applied Thermal Engineering*, 2006.

[109] M. Toub, C. R. Reddy, M. Razmara, M. Shahbakhti, R. D. Robinett III, and G. Aniba, "Model-based predictive control for optimal microcsp operation integrated with building hvac systems," *Energy Conversion and Management*, vol. 199, p. 111 924, 2019.

[110] D. B. Crawley, L. K. Lawrie, C. O. Pedersen, and F. C. Winkelmann, "Energy plus: Energy simulation program," *ASHRAE journal*, vol. 42, no. 4, pp. 49–56, 2000.

[111] Z. Du, B. Fan, J. Chi, and X. Jin, "Sensor fault detection and its efficiency analysis in air handling unit using the combined neural networks," *Energy and Buildings*, vol. 72, pp. 157–166, 2014.

[112] Z. Du, B. Fan, X. Jin, and J. Chi, "Fault detection and diagnosis for buildings and hvac systems using combined neural networks and subtractive clustering analysis," *Building and Environment*, vol. 73, pp. 1–11, 2014.

[113] S. Wang and J. Cui, “Sensor-fault detection, diagnosis and estimation for centrifugal chiller systems using principal-component analysis method,” *Applied Energy*, vol. 82, no. 3, pp. 197–213, 2005.

[114] J. Liu, M. Zhang, H. Wang, W. Zhao, and Y. Liu, “Sensor fault detection and diagnosis method for ahu using 1-d cnn and clustering analysis,” *Computational intelligence and neuroscience*, vol. 2019, 2019.

[115] M. S. Mirnaghi and F. Haghighat, “Fault detection and diagnosis of large-scale hvac systems in buildings using data-driven methods: A comprehensive review,” *Energy and Buildings*, p. 110 492, 2020.

[116] V. Reppa, P. Papadopoulos, M. M. Polycarpou, and C. G. Panayiotou, “A distributed architecture for hvac sensor fault detection and isolation,” *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1323–1337, 2014.

[117] J. C. da Silva, A. Saxena, E. Balaban, and K. Goebel, “A knowledge-based system approach for sensor fault modeling, detection and mitigation,” *Expert Systems with Applications*, vol. 39, no. 12, pp. 10 977–10 989, 2012.

[118] J. Fonollosa, A. Vergara, and R. Huerta, “Algorithmic mitigation of sensor failure: Is sensor replacement really necessary?” *Sensors and Actuators B: Chemical*, vol. 183, pp. 211–221, 2013.

[119] W. Kim and S. Katipamula, “A review of fault detection and diagnostics methods for building systems,” *Science and Technology for the Built Environment*, vol. 24, no. 1, pp. 3–21, 2018.

[120] Z.-H. Zhou, “A brief introduction to weakly supervised learning,” *National science review*, vol. 5, no. 1, pp. 44–53, 2018.

[121] Y.-Y. Sun, Y. Zhang, and Z.-H. Zhou, “Multi-label learning with weak label,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, 2010.

[122] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille, “Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1742–1750.

[123] X. Zhu and A. B. Goldberg, “Introduction to semi-supervised learning,” *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[124] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," *arXiv preprint arXiv:1606.07947*, 2016.

[125] D. A. Hashimoto, G. Rosman, D. Rus, and O. R. Meireles, "Artificial intelligence in surgery: Promises and perils," *Annals of surgery*, vol. 268, no. 1, pp. 70–76, 2018.

[126] R. Rendall, I. Castillo, B. Lu, *et al.*, "Image-based manufacturing analytics: Improving the accuracy of an industrial pellet classification system using deep neural networks," *Chemometrics and Intelligent Laboratory Systems*, vol. 180, pp. 26–35, 2018.

[127] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko, "Semi-supervised learning with ladder networks," *arXiv preprint arXiv:1507.02672*, 2015.

[128] W. Dong-DongChen and Z.-H. WeiGao, "Tri-net for semi-supervised deep learning," in *International Joint Conferences on Artificial Intelligence*, 2018.

[129] J. Li, R. Socher, and S. C. Hoi, "Dividemix: Learning with noisy labels as semi-supervised learning," in *International Conference on Learning Representations*, 2019.

[130] R. Li, Q. Jiao, W. Cao, H.-S. Wong, and S. Wu, "Model adaptation: Unsupervised domain adaptation without source data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9641–9650.

[131] J. Dong, Y. Cong, G. Sun, B. Zhong, and X. Xu, "What can be transferred: Unsupervised domain adaptation for endoscopic lesions segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4023–4032.

[132] G. Wilson and D. J. Cook, "A survey of unsupervised deep domain adaptation," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 5, pp. 1–46, 2020.

[133] J. Dong, Y. Cong, G. Sun, and D. Hou, "Semantic-transferable weakly-supervised endoscopic lesions segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 10 712–10 721.

[134] R. Kiryo, G. Niu, M. C. d. Plessis, and M. Sugiyama, "Positive-unlabeled learning with non-negative risk estimator," *arXiv preprint arXiv:1703.00593*, 2017.

[135] X. Chen, W. Chen, T. Chen, *et al.*, "Self-pu: Self boosted and calibrated positive-unlabeled training," in *International Conference on Machine Learning*, PMLR, 2020, pp. 1510–1519.

[136] F. Perez, R. Lebret, and K. Aberer, "Weakly supervised active learning with cluster annotation," *arXiv preprint arXiv:1812.11780*, 2018.

[137] S. Belharbi, I. Ben Ayed, L. McCaffrey, and E. Granger, "Deep active learning for joint classification & segmentation with weak annotator," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 3338–3347.

[138] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari, "Learning with noisy labels.," in *NIPS*, vol. 26, 2013, pp. 1196–1204.

[139] A. Ghosh, N. Manwani, and P. Sastry, "Making risk minimization tolerant to label noise," *Neurocomputing*, vol. 160, pp. 93–107, 2015.

[140] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 38, no. 3, pp. 447–461, 2015.

[141] Y. Xu, Y. Wang, H. Chen, *et al.*, "Positive-unlabeled compression on the cloud," *arXiv preprint arXiv:1909.09757*, 2019.

[142] M. R. Loghmani, M. Vincze, and T. Tommasi, "Positive-unlabeled learning for open set domain adaptation," *Pattern Recognition Letters*, vol. 136, pp. 198–204, 2020.

[143] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 4080–4090.

[144] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, pp. 1–54, 2019.

[145] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 935–942.

[146] S. Pouyanfar, Y. Tao, A. Mohan, *et al.*, "Dynamic sampling in convolutional neural networks for imbalanced data classification," in *2018 IEEE conference on multimedia information processing and retrieval (MIPR)*, IEEE, 2018, pp. 112–117.

[147] T. Cao, M. Law, and S. Fidler, "A theoretical analysis of the number of shots in few-shot learning," *arXiv preprint arXiv:1909.11722*, 2019.

[148] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *International Conference on Learning Representations*, 2019.

[149] E. Triantafillou, T. Zhu, V. Dumoulin, *et al.*, "Meta-dataset: A dataset of datasets for learning to learn from few examples," in *International Conference on Learning Representations*, 2019.

[150] L. Wang, S. Xu, X. Wang, and Q. Zhu, "Addressing class imbalance in federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 10 165–10 173.

[151] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, "Exploring generalization in deep learning," in *Advances in neural information processing systems*, 2017, pp. 5947–5956.

[152] R. Amit and R. Meir, "Meta-learning by adjusting priors based on extended pac-bayes theory," in *International Conference on Machine Learning*, PMLR, 2018, pp. 205–214.

[153] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020.

[154] D. A. McAllester, "Some pac-bayesian theorems," *Machine Learning*, vol. 37, no. 3, pp. 355–363, 1999.

[155] P. Germain, F. Bach, A. Lacoste, and S. Lacoste-Julien, "Pac-bayesian theory meets bayesian inference," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 1884–1892.

[156] D. McAllester, "Simplified pac-bayesian margin bounds," in *Learning theory and Kernel machines*, Springer, 2003, pp. 203–215.

[157] B. Neyshabur, S. Bhojanapalli, and N. Srebro, "A pac-bayesian approach to spectrally-normalized margin bounds for neural networks," in *International Conference on Learning Representations*, 2018.

[158] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation: Learning bounds and algorithms," *arXiv preprint arXiv:0902.3430*, 2009.

[159] S. Kassam, "Quantization based on the mean-absolute-error criterion," *IEEE Transactions on Communications*, vol. 26, no. 2, pp. 267–270, 1978.

[160] R. F. Gunst and R. L. Mason, “Biased estimation in regression: An evaluation using mean squared error,” *Journal of the American Statistical Association*, vol. 72, no. 359, pp. 616–628, 1977.

[161] C. Yi and J. Huang, “Semismooth newton coordinate descent algorithm for elastic-net penalized huber loss regression and quantile regression,” *Journal of Computational and Graphical Statistics*, vol. 26, no. 3, pp. 547–557, 2017.

[162] R. Koenker, “Quantile regression for longitudinal data,” *Journal of Multivariate Analysis*, vol. 91, no. 1, pp. 74–89, 2004.

[163] Y. Luo, Z. Wang, Z. Huang, and M. Baktashmotlagh, “Progressive graph learning for open-set domain adaptation,” in *International Conference on Machine Learning*, PMLR, 2020, pp. 6468–6478.

[164] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: Maximizing for domain invariance,” *arXiv preprint arXiv:1412.3474*, 2014.

[165] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” 2011.

[166] L. Deng, “The mnist database of handwritten digit images for machine learning research [best of the web],” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[167] J. J. Hull, “A database for handwritten text recognition research,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[168] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, “Visda: The visual domain adaptation challenge,” *arXiv preprint arXiv:1710.06924*, 2017.

[169] A. Krizhevsky, G. Hinton, *et al.*, “Learning multiple layers of features from tiny images,” 2009.

[170] T. Kim and C. Kim, “Attract, perturb, and explore: Learning a feature alignment network for semi-supervised domain adaptation,” in *European Conference on Computer Vision*, Springer, 2020, pp. 591–607.

[171] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, “Semi-supervised domain adaptation via minimax entropy,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8050–8058.

[172] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Proceedings of the 17th International Conference on Neural Information Processing Systems*, 2004, pp. 529–536.

[173] R. Ranjan, C. D. Castillo, and R. Chellappa, "L2-constrained softmax loss for discriminative face verification," *arXiv preprint arXiv:1703.09507*, 2017.

[174] C.-W. Lee, W. Fang, C.-K. Yeh, and Y.-C. F. Wang, "Multi-label zero-shot learning with structured knowledge graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1576–1585.

[175] J. Lu, L. Du, M. Liu, and J. Dipnall, "Multi-label few/zero-shot learning with knowledge aggregated from multiple label graphs," *arXiv preprint arXiv:2010.07459*, 2020.

[176] A. Ben-Cohen, N. Zamir, E. Ben-Baruch, I. Friedman, and L. Zelnik-Manor, "Semantic diversity learning for zero-shot multi-label classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 640–650.

[177] S. Liu, L. Zhang, X. Yang, H. Su, and J. Zhu, "Query2label: A simple transformer way to multi-label classification," *arXiv preprint arXiv:2107.10834*, 2021.

[178] T. Ridnik, G. Sharir, A. Ben-Cohen, E. Ben-Baruch, and A. Noy, "Ml-decoder: Scalable and versatile classification head," *arXiv preprint arXiv:2111.12933*, 2021.

[179] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[180] A. Zareian, K. D. Rosa, D. H. Hu, and S.-F. Chang, "Open-vocabulary object detection using captions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 393–14 402.

[181] Y. Du, F. Wei, Z. Zhang, M. Shi, Y. Gao, and G. Li, "Learning to prompt for open-vocabulary object detection with vision-language model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 084–14 093.

[182] G. Ghiasi, X. Gu, Y. Cui, and T.-Y. Lin, "Open-vocabulary image segmentation," *arXiv preprint arXiv:2112.12143*, 2021.

[183] H. Yang, J. Tianyi Zhou, Y. Zhang, B.-B. Gao, J. Wu, and J. Cai, "Exploit bounding box annotations for multi-label object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 280–288.

[184] Z. Wang, T. Chen, G. Li, R. Xu, and L. Lin, "Multi-label image recognition by recurrently discovering attentional regions," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 464–472.

[185] B.-B. Gao and H.-Y. Zhou, "Learning to discover multi-class attentional regions for multi-label image recognition," *IEEE Transactions on Image Processing*, vol. 30, pp. 5920–5932, 2021.

[186] R. You, Z. Guo, L. Cui, X. Long, Y. Bao, and S. Wen, "Cross-modality attention with semantic graph embedding for multi-label classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 12 709–12 716.

[187] T. Chen, M. Xu, X. Hui, H. Wu, and L. Lin, "Learning semantic-specific graph representation for multi-label image recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 522–531.

[188] M. Shi, Y. Tang, X. Zhu, and J. Liu, "Multi-label graph convolutional network representation learning," *IEEE Transactions on Big Data*, 2020.

[189] J. Ye, J. He, X. Peng, W. Wu, and Y. Qiao, "Attention-driven dynamic graph convolutional network for multi-label image recognition," in *European conference on computer vision*, Springer, 2020, pp. 649–665.

[190] M. Norouzi, T. Mikolov, S. Bengio, *et al.*, "Zero-shot learning by convex combination of semantic embeddings," *arXiv preprint arXiv:1312.5650*, 2013.

[191] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for image classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 7, pp. 1425–1438, 2015.

[192] Y. Zhang, B. Gong, and M. Shah, "Fast zero-shot image tagging," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2016, pp. 5985–5994.

[193] J.-H. Kim, J. Jun, and B.-T. Zhang, "Bilinear attention networks," *Advances in neural information processing systems*, vol. 31, 2018.

[194] D. Huynh and E. Elhamifar, "A shared multi-attention framework for multi-label zero-shot learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8776–8786.

[195] S. Narayan, A. Gupta, S. Khan, F. S. Khan, L. Shao, and M. Shah, "Discriminative region-based multi-label zero-shot learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8731–8740.

[196] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "Visualbert: A simple and performant baseline for vision and language," *arXiv preprint arXiv:1908.03557*, 2019.

[197] G. Li, N. Duan, Y. Fang, M. Gong, and D. Jiang, "Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 11 336–11 344.

[198] W. Kim, B. Son, and I. Kim, "Vilt: Vision-and-language transformer without convolution or region supervision," in *International Conference on Machine Learning*, PMLR, 2021, pp. 5583–5594.

[199] N. Mu, A. Kirillov, D. Wagner, and S. Xie, "Slip: Self-supervision meets language-image pre-training," *arXiv preprint arXiv:2112.12750*, 2021.

[200] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," *arXiv preprint arXiv:2201.12086*, 2022.

[201] J. Yang, J. Duan, S. Tran, *et al.*, "Vision-language pre-training with triple contrastive learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 671–15 680.

[202] M. Gao, C. Xing, J. C. Niebles, *et al.*, "Open vocabulary object detection with pseudo bounding-box labels," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part X*, Springer, 2022, pp. 266–282.

[203] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary object detection via vision and language knowledge distillation," *arXiv preprint arXiv:2104.13921*, 2021.

[204] M. A. Bravo, S. Mittal, and T. Brox, "Localized vision-language matching for open-vocabulary object detection," in *Pattern Recognition: 44th DAGM German Conference, DAGM GCPR*

2022, Konstanz, Germany, September 27–30, 2022, Proceedings, Springer, 2022, pp. 393–408.

[205] W. Kuo, Y. Cui, X. Gu, A. Piergiovanni, and A. Angelova, "F-vlm: Open-vocabulary object detection upon frozen vision and language models," *arXiv preprint arXiv:2209.15639*, 2022.

[206] C. Ma, Y. Yang, Y. Wang, Y. Zhang, and W. Xie, "Open-vocabulary semantic segmentation with frozen vision-language models," *arXiv preprint arXiv:2210.15138*, 2022.

[207] G. Ghiasi, X. Gu, Y. Cui, and T.-Y. Lin, "Scaling open-vocabulary image segmentation with image-level labels," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVI*, Springer, 2022, pp. 540–557.

[208] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "Nus-wide: A real-world web image database from national university of singapore," in *Proceedings of the ACM international conference on image and video retrieval*, 2009, pp. 1–9.

[209] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[210] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[211] T. Ridnik, E. Ben-Baruch, A. Noy, and L. Zelnik-Manor, "Imagenet-21k pretraining for the masses," *arXiv preprint arXiv:2104.10972*, 2021.

[212] E. Ben-Baruch, T. Ridnik, N. Zamir, *et al.*, "Asymmetric loss for multi-label classification," *arXiv preprint arXiv:2009.14119*, 2020.

[213] A. Gupta, S. Narayan, S. Khan, F. S. Khan, L. Shao, and J. van de Weijer, "Generative multi-label zero-shot learning," *arXiv preprint arXiv:2101.11606*, 2021.

[214] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, "Multi-label image recognition with graph convolutional networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5177–5186.

[215] Y. Liu, L. Sheng, J. Shao, J. Yan, S. Xiang, and C. Pan, "Multi-label image classification via knowledge distillation from weakly-supervised detection," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 700–708.

[216] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[217] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[218] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[219] Y. Liu, M. Ott, N. Goyal, *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[220] Y. Sun, S. Ma, R. Madaan, R. Bonatti, F. Huang, and A. Kapoor, "Smart: Self-supervised multi-task pretraining with control transformers," *arXiv preprint arXiv:2301.09816*, 2023.

VITA

Shichao Xu was born in 1996 in China. He received his undergraduate degree from Shanghai Jiao Tong University (ACM Honors Class) in 2018, advised by Prof. Liqing Zhang in the Center for Brain-like Computing and Machine Intelligence. He received the Ph.D. degree from Northwestern University in 2023. His research works focus on learning from limited or imperfect data, and explore the area of computer vision, multi-model, and control applications using machine learning techniques under the practical settings.