**Master's Thesis of College of Engineering**

# Explainable reinforcement learning and rule reduction for advanced building control

강화학습을 적용한 실용적인 건물 시스템 제어

**August 2021**

**Department of Architecture and Architectural Engineering**

**College of Engineering**
**Seoul National University**

**Seongkwon Cho**

# Explainable reinforcement learning and rule reduction for advanced building control

Advised by Prof. Park, Cheol-Soo

Submitting a master's thesis of College of Engineering

July 2021

Department of Architecture and Architectural Engineering
College of Engineering, Seoul National University

Seongkwon Cho

Confirming the master's thesis written by
Seongkwon Cho

August 2021

| | |
|---|---|
| Chair | Prof. Myoung Souk Yeo |
| Vice Chair | Prof. Cheol-Soo Park |
| Examiner | Prof. Sunsook Kim |

# Abstract

Building controls are becoming complicated because modern building systems must respond to not only conventional systems like HVAC and lighting, but also to novel systems such as intermittent renewables, energy storage systems, and more. Therefore, the advanced building controllers must balance the trade-off between multiple objectives and automatically adapt to dynamic environment. Although it is widely acknowledged that reinforcement learning (RL) can be beneficially used for better building control, there are several challenges that should be addressed for real life application of RL: (1) unstable and poor control actions during early training period of RL may cause unexpected costs; (2) many RL-based control actions still remain unexplainable for daily practice of facility managers. By applying RL algorithms as artificial intelligences that are the subject of decision-making, owners and operators of buildings need to be reassured about the controllers' intentions.

To address the first challenge, federated model, a novel concept of simulation model, is proposed for pre-training RL agents. The federated model is an integrated data-driven model that divides a building system into several modules based on physical causality and develops each module into a data-driven model to perform simulations on building systems. A federated model of a complex cooling system of a target building is realized using six modules, each developed using data gathered from BEMS. By developing the federated model, limitations of physics-based simulation models (eg. topology rules, model calibration) are overcome. Deep Q-network (DQN) is applied to learn the dynamics of the cooling system and explore control strategies that can reduce energy use while providing cold for the building. By comparing the control performance of DQN with the performance of baseline control, it is shown that RL controller can significantly enhance control efficiency of the system and the federated model can provide sufficient virtual experience for the controller.

To enhance interpretability of the DQN agent, decision tree is used to extract explanation of the decision making process of the agent. State-action pairs generated

by the agent is used train a decision tree. Post-hoc interpretation using a shallow but easily interpretable model enhances transparency and interpretability of reinforcement learning. Also, the result of classification made by the decision tree provides 'If-then' rules which are reduced version of control strategies made by the artificial intelligence. The performance of the reduced rule-based control is also compared to the performance of DQN controller. It is demonstrated that the reduced rule is good-enough and the difference in energy savings between the two is marginal, resulting in 2.8%.

This study reports the development of explainable RL for cooling control of an existing office building. A decision tree is applied to trained DQN agent and then a set of reduced-order control rules are suggested. This study proposes rule reduction framework using explainable reinforcement learning and demonstrates that reduced rules can perform as well as complex reinforcement learning algorithms. The significance of this study lies in proposing how to derive rules with quantitative evaluation for building control.

**Keyword :** explainable reinforcement learning, rule reduction, reduced rule-based control, DQN

**Student Number :** 2019-21675

# Contents

# List of Figures

# List of Tables

# 1. Introduction

Buildings consume about 40% of total primary energy in countries like United States. From a life cycle perspective, most of a building's energy is consumed during its operational phase, thus, it is crucial to identify and evaluate impactful energy-saving technologies and strategies during this phase (Hong et. al., 2018). Moreover, in developed countries the buildings sector is dominated by existing buildings. Accordingly, improvement of existing building operation can be a key strategy for reduction the overall energy use of the buildings. Systems in buildings, such as lighting, windows, and HVAC equipment have been equipped with smart controllers and these systems may open opportunities for improving building efficiency (Wang & Hong, 2020). Well-designed control systems can increase building efficiency up to 30% without the need to upgrade existing appliances (DOE, 2015).

## 1.1 Control of building systems

Majority of HVAC systems are controlled using rule-based approach based on the building operators' experience and knowledge. The rule-based approach is simple and easy to apply but, as quantitative and scientific evaluation for the approach is not made, it is hard to be optimal and its performance can be inferior to that of other control methods.

To address the aforementioned problem of rule-based control, Model-Predictive Control (MPC) has attracted attention. MPC is an adaptive control approach that uses simulation models to predict the future states of buildings or systems and make control strategies based on the predictions (Afram & Janabi-Sharifi, 2014). The goal to be achieved through control is expressed as a cost function and constraints, such as limitation of control variables or satisfaction range for state variables, can be defined. Although the performance of MPC is promising, MPC has not yet been widely adopted by the building industry. This is because it is labor-intensive to develop simulation models with high fidelity and requires expertise to use (Wang & Hong, 2020).

Recently, reinforcement learning-based control of building systems has been attracting significant attention in building simulation domain. Reinforcement learning (RL) is a kind of machine learning that explores optimal and sequential decision-making strategies to achieve goals and it is being actively applied on various fields from self-driving to robotics. Building controls are becoming complicated because modern building systems must respond to not only conventional systems like HVAC and lighting, but also to novel systems such as intermittent renewables, energy storage systems, and more. Therefore, the advanced building controllers must balance the trade-off between multiple objectives and automatically adapt to environment. The controllers based on RL have potential to achieve these performances. As the RL controllers continuously learn from different operating conditions through interaction with building, they are capable of adapting to the local environment and operation conditions. Although many simulation studies have been conducted and proved the potential of RL, there are a limited number of studies on its practical implementation and evaluation. To further understand the opportunities and challenges of reinforcement learning based building system control, implementation studies should be conducted using existing buildings or data gathered from them.

## 1.2 Problem Description

An agent, a decision maker of RL algorithm, directly interacts with building environment in trial-error manner to learn the dynamics of the environment and find optimal control strategies to control them. Thus, many studies on RL-based building control system says RL has its own strength since it does not require any sophisticated mathematical simulation model compared to MPC. Though those simulation studies proved that RL can make well-considered strategies to optimize energy efficiency of the building environment, still application of RL-based control system has significant challenges in REAL buildings:

- RL may cause unexpected costs with unstable and poor control actions during its own early training period on real building situation (Wang & Hong, 2020). While those periods, it will study how to make optimal strategy by testing new decisions and evaluating the outcomes, and it is possible to fail

to maintain thermal comfort of building occupants or to cause system breakdown (which causes tremendous financial damage on building maintenance).

- Another challenge for implementing RL for real-world problems is lack of "interpretability" and "reliability" (Dulac-Arnold et. al., 2019). As RL agents are neural network models in form of black boxes, it is hard to understand and explain their algorithm of decision making. By applying RL algorithms as artificial intelligences that are the subject of decision-making, owners and operators of buildings need to be reassured about the controllers' intentions. The ability to provide explanations of the behavior of these artificial agents is particularly important in scenarios where humans need to collaborate with them.

Thus, study how to avoid undesirable outcomes, guarantee control robustness, and secure interpretability and reliability are key points of implementing RL in real buildings.

One way to address the problem is pre-training agents using simulators. Through pre-interaction with virtual environments, agents can be guided to explore optimal control strategies to achieve goals before they are implemented into the actual environment. Through this practice, RL agents can minimize their mistakes on real situation. The virtual environments can be generated in two approaches: physics-based simulation model and data-driven model. Physics-based simulation models can perform sophisticated simulations of thermal dynamics of buildings based on detailed input variables. However, most simulation tools have predefined topology rules for computational efficiency, which may not allow accurate modeling of systems in real buildings. Besides, to reduce the performance gap between simulation results and measurements, model calibration is essential, and it is usually time-consuming and sometimes impossible to perform because of insufficient data. Therefore, if BEMS (Building Energy Management System) data is available, it is practical and rational to generate virtual environment using data-driven methods.

Post-hoc interpretation using shallow but easily interpretable models, beside the black box models, is one way to enhance transparency and interpretability of reinforcement learning (Alharin et. al., 2018). “Reliability” is not built up simply by having the performance of reducing energy use while not messing up the system control. It is built up when the decision-making process of artificial agents is rationale and human understandable. Decision trees are intuitive and easily explainable machine learning algorithm that are widely used to extract classification rules. They are also one of the most popular method to distal explanation from RL agent models (Madumal et. al., 2020). As agents of RL learn how to map states to actions, the trained artificial intelligences generate state-action pairs with their own decision-making rules. The decision-making rules can be interpreted by classifying those state-action data using decision trees. The generated explanation models have the potential to provide intuitive and natural explanations, allowing the human (eg. building operators) a deeper understanding of the agents and to build “trust”. Also, the result of classification can provide ‘If-then’ rules which can be implemented to real building system control problems with minimum cost. The derived rules are expected to be reduced version of complex control strategies made by artificial intelligences.

## 1.3 Goal

The objective of this study is to propose a practical framework of implementing reinforcement learning based optimal control to an existing building (Figure 1-1). If data gathered from an existing building is available, it can be practical to give artificial intelligences opportunities to learn the building dynamics using the data. In this study, data-driven models are constructed for a complicated cooling system of target building, which consists of ice-based thermal storage with two chillers and 18 geo-thermal heat pumps. The models are used to provide pre-training phase for an agent of deep Q-learning, one of the most popular algorithms of reinforcement learning. As the trained artificial intelligence could not be implemented to the real system, the performance of it is proved using the model. Furthermore, a novel concept of interpreting RL agents is proposed. Providing explanation of decision-making process of agents can enhance the reliability of RL algorithms. Also, considering the nature of

building system control with high costs incurred by undesirable results, reduced rules that are generated using decision trees can be a realistic alternative to direct implementation of RL algorithms. It is expected that this could provide a vision for collaboration of artificial intelligence and humans in building control.

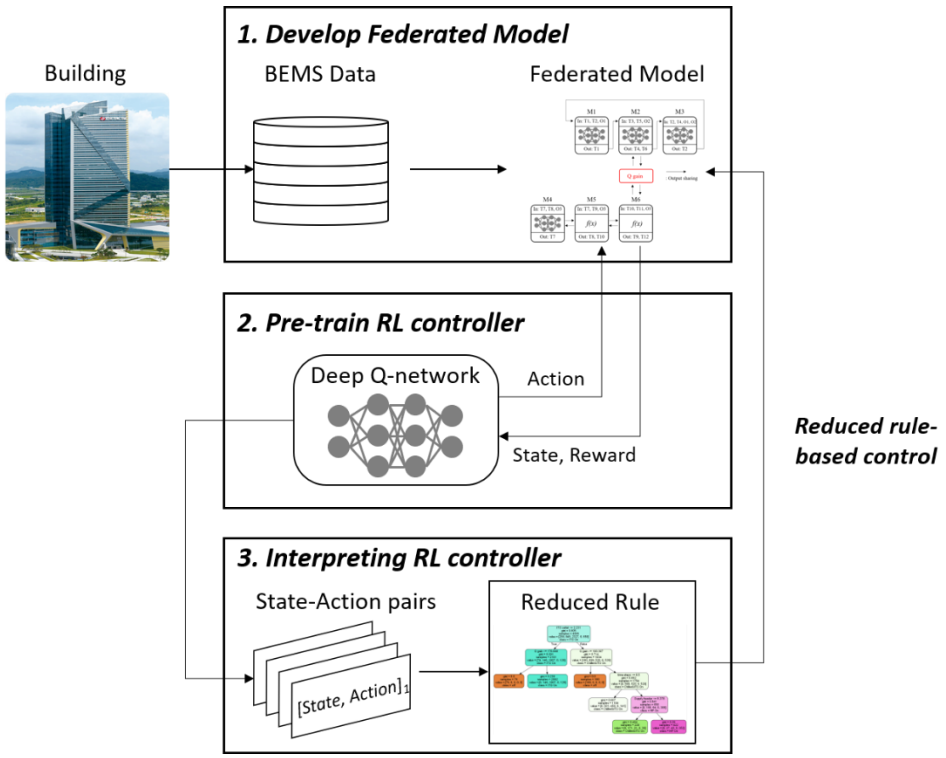



**Figure 1-1. Rule reduction for control of existing building**

## 1.4 Thesis Outline

The research presented in this thesis concerns the field of optimal control of building system. The aim is to propose practical approach of implementing reinforcement learning based optimal control and a novel concept of interpreting reinforcement learning agents.

The content of the following chapters is as follows. Chapter II reviews previous studies on reinforcement learning based optimal control of building system and theory of deep reinforcement learning algorithm; it provides a background on reinforcement learning. Chapter III presents the necessity of interpreting reinforcement learning agents with introducing a method of distilling explanation using decision trees. Chapter IV presents overall implementing methodology of the proposed framework and the structure of a simulation model developed to realize the system of the target building is introduced. A federated model which consists of several data-driven models is introduced and the usefulness as a simulator is also discussed. Chapter V analyzes the implementation result of a reinforcement learning based controller to the target building. Also, interpretation process of decision-making rules of reinforcement learning agents are shown. Finally, Chapter VI completes thesis by providing a summary of the work, limitations of the study, and future challenges.

# 2. Deep Q-network (DQN)

This chapter gives an overview of a reinforcement learning algorithm, especially, deep Q-learning that is one of the most popular deep reinforcement learning algorithms which is applied for the study. Deep Q-learning (DQL) is a model-free reinforcement learning algorithm which does not require mathematical simulation models while exploring optimal control strategies of systems. Instead, the DQL uses experience data gathered directly from systems to learn the dynamics of systems and establish control strategies. However, applying the model-free approach reinforcement learning algorithm directly to real-world control problems has several challenges and they are discussed in this chapter. Also, earlier works on implementing reinforcement learning for control systems of existing buildings and findings are introduced.

## 2.1. Summary of reinforcement learning

Reinforcement learning is one of machine learning paradigms that learns how to map observations to actions (Sutton and Barto, 2018). Artificial decision makers of reinforcement learning attempt various actions to learn strategies that meet the purpose and collect the result data which can be called experience. The decision makers repeat the attempt to get better results, and learn optimal actions to achieve the goal from those attempts. The way reinforcement learning learns is similar to how human learns and this is why it is called artificial intelligence. In this section, elements and basic theories of reinforcement learning are introduced.

### 2.1.1 Elements of reinforcement learning

A decision maker of reinforcement learning is called an agent and the entity that interacts with the agent is called the environment. The agent observes state ($s_t$) from environment and determines an action ($a_t$) according to the state. The state of environment changes due to the action and the changed state ($s_{t+1}$) is passed to the agent with reward ($r_t$) for the action. Reinforcement learning problem is a sequential decision-making problem in which agents choose actions to maximize rewards.

The discrete time-step is represented as $t$ and $s_t \in S$ represents the state of each time-step where $S$ represents the entire possible states. $a_t \in A(s_t)$ represents action at time-step $t$ where $A(s_t)$ represents all possible actions that agents can take on $s_t$. The rewards, $r_t \in R$, are represented by numbers and define the goal in a reinforcement learning problem. A policy defines the probability of choosing an action for the agent in certain states. $\pi_t(s,a)$ represents the policy of time-step $t$ and it means the probability of taking action $a$ in state $s$. The agent takes action based on the policy and updates the policy during the learning period to find optimal policies.

Reinforcement learning uses the concept of Markov decision process (MDP), discrete time stochastic control system. The agent chooses action only based on the current state, which is possible based on the premise that the state has Markov property. Markov property is expressed as Equation 2-1: The probability of taking an action ($a_t$) only based on current state ($s_t$) is same as the probability of taking the action based on all the information from the past $(s_t, a_{t-1}, s_{t-1}, \cdots, a_0, s_0)$. Reinforcement learning has shown that one time-step dynamics decision making is as effective as making decisions based on all past information.

$$\Pr(a_t|s_t) = \Pr(a_t|s_t, a_{t-1}, s_{t-1}, \cdots, a_0, s_0) \quad \text{(Equation 2-1)}$$

In MDP (Figure 2-1), the relationships between elements such as states, actions, and reward are expressed stochastically. At each time step, the process is in some state $s_t$ and the agent chooses action that is possible in $s_t$. By the action, the state changes to the next state $s_{t+1}$ stochastically and the agent get the corresponding reward, $R(s_{t+1}|s_t, a_t)$. Transition probability, $\Pr(s_{t+1}|s_t, a_t)$, represents the probability of change from the current state ($s_t$) to next state ($s_{t+1}$) when taking an action ($a_t$). In other words, the next state depends on only current state and current action with given $s_t$ and $a_t$, which means MDP satisfy the Markov property.

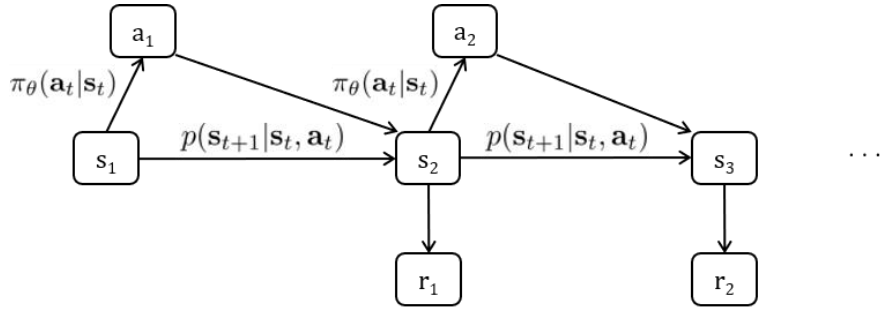



**Figure 2-1. Markov decision process (Kim, 2021)**

Return (Equation 2-2) represents the cumulative reward earned by the agent during Markov decision process. The discount rate, $\gamma \in [0,1]$, essentially determines how much agents cares about future rewards relative to immediate rewards. For example, $\gamma = 0$ is to explore strategies to maximize only the current reward and $\gamma = 1$ means that all future rewards will be considered equally. Especially for infinite Markov decision process, the discount rate prevents returns from diverging.

$$R_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k} \quad \text{(Equation 2-2)}$$

### 2.1.2 Value function

Agents of reinforcement learning earn rewards through interaction with environment and develop policies based on rewards. The goal of reinforcement learning is to maximize return, the cumulative rewards earned during the interaction, and a value function is introduced to estimate the return probabilistically. The reason that value functions are expressed as expected values is that they reflect future return that have not yet been experienced. The expectations can be updated by the interaction of the agent and the environment.

Two different value functions are defined: state value functions and action value functions. State value function (Equation 2-3) of a state $s_t$ under a policy π is the expected return when starting in $s_t$ and following the policy π, where τ denotes the trajectory (at, st+1, at+1, $\cdots, s_{T-1}, a_{T-1}, s_T$) that occurs according to the policy. There are various states in which state $s_t$ can change at time t+1, and the change occurs following the transition probability, mentioned above. In other words, there are a number of cases that can reach the goal in a specific state, $s_t$, and accordingly, there are many cases of return value. The state value function is the expected value of the return for all these cases.

Action value function (Equation 2-4) is the expected return from the action a taken in state $s_t$ under at policy π, where τ denotes the trajectory ($s_{t+1}$, $a_{t+1}$, $\cdots, s_{T-1}, a_{T-1}, s_T$) that occurs according to the policy. The action value function defines a value for every action that can be taken in a particular state. The state value of $s_t$ is the mean value of the action values of all actions that can be taken in $s_t$ (Equation 2-5).

$$V^{\pi}(s_t) = E_{\tau \sim p(\tau|s_t)}[\sum_{k=t}^{T} \gamma^{k-t} \cdot r(s_k, a_k)|s_t] \quad \text{(Equation 2-3)}$$

$$Q^{\pi}(s_t, a_t) = E_{\tau \sim p(\tau|s_t, a_t)}[\sum_{k=t}^{T} \gamma^{k-t} \cdot r(s_k, a_k)|s_t, a_t] \quad \text{(Equation 2-4)}$$

$$V^{\pi}(s_t) = E_{a_t \sim \pi}[Q^{\pi}(s_t, a_t)] \quad \text{(Equation 2-5)}$$

The relationship between the current state value and the state value of the next time-step follows Equation 2-6, which is called the Bellman equation. The relationship between the current action value and the action value of next time-step can also be expressed by the Bellman equation (Equation 2-7).

$$V^{\pi}(s_t) = E_{a_t \sim \pi}[r(s_t, a_t) + E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[\gamma \cdot V^{\pi}(s_{t+1})]] \quad \text{(Equation 2-6)}$$

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[E_{a_t \sim \pi}[\gamma \cdot Q^{\pi}(s_{t+1}, a_{t+1})]] \quad \text{(Equation 2-7)}$$

The policy that maximizes the state value is defined as optimal policy. The state value function and the action value function when the optimal policy is applied are defined as the optimal state value function (Equation 2-8) and the optimal action value function (Equation 2-9), respectively, and these are called the Bellman optimal equations.

$$V^{*}(s_t) = \max_{a_t}[r(s_t, a_t) + E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[\gamma \cdot V^{*}(s_{t+1})]] \quad \text{(Equation 2-8)}$$

$$Q^{*}(s_t, a_t) = r(s_t, a_t) + E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[\gamma \cdot \max_{a_{t+1}} Q^{*}(s_{t+1}, a_{t+1})] \quad \text{(Equation 2-9)}$$

Equation 2-10 shows the relationship between the optimal state value function and the optimal action value function: optimal state value can be obtained by maximizing optimal action value. As mentioned above, optimal policy is the policy that maximizes state value, so the optimal policy is to take action with the largest action value in each state (Equation 2-11) which maximizes the return.

$$V^{*}(s_t) = \max_{a_t} Q^{*}(s_t, a_t) \quad \text{(Equation 2-10)}$$

$$\pi^{*}(a_t|s_t) = \operatorname{argmax}_{a_t} Q^{*}(s_t, a_t) \quad \text{(Equation 2-11)}$$

## 2.2. Deep Q-learning

### 2.2.1 Temporal difference (TD) learning and Q-learning

Dynamic programming algorithms are of limited utility in reinforcement learning because of their assumption of a perfect model. To compensate for these shortcomings, the Monte Carlo method, an experience-based reinforcement learning concept, is proposed. Monte Carlo methods use experience sample sequences of states, actions, and rewards to average returns for each state-action pair and approximates it as a state value function of the state.

The approximated state value function is expressed as Equation 2-12, where $\mathrm{V(s)}$ denotes approximated state value function in state $s$; n means the number of times the $s$ has been visited over episodes; $R_i(s)$ denotes the return obtained by visiting the $s$ of i[th] episodes. State value functions for all states can be approximated in the same way.

$$\mathrm{V(s)} = \frac{1}{n}\sum_{i=1}^{n} R_i(s) \qquad \text{(Equation 2-12)}$$

Equation 2-13 shows how to update the value function for each episode by approximating the state value function where, $V(s)_n$ denotes nth updated state value function. $\sum_{i=1}^{n} R(s)_i$ can be separated into the return of the nth episode ($R(s)_n$) and the approximated state value function of the n-1th episode ($V(s)_{n-1}$). In other words, the nth update of approximated state value function of state s is performed by reflecting the difference between $V(s)_{n-1}$ and $R(s)_n$ by a weight of 1/n. Equation 2-14 shows 1/n in $\alpha$ as a concept of weight which means it is possible to derive an optimal state value function of state s without a model of the system using Monte Carlo method. However, as Monte Carlo method takes into account entire return gained until the episode is over, it is still of limited utility for real world problems.

$$V(s)_n = \frac{1}{n}\sum_{i=1}^{n} R_i(s) \quad \text{(Equation 2-13)}$$
$$= \frac{1}{n}\left(R_n(s) + \sum_{i=1}^{n-1} R_i(s)\right)$$
$$= \frac{1}{n}(R_n(s) + (n-1)V(s)_{n-1})$$
$$= V(s)_{n-1} + \frac{1}{n}(R_n(s) - V(s)_{n-1})$$

$$V(s_t) \leftarrow V(s_t) + \alpha(R_t - V(s_t)) \quad \text{(Equation 2-14)}$$

Unlike Monte Carlo method, Temporal Difference (TD) learning is a reinforcement learning method that updates the state value function of state $s$ based on samples of experience rather than waiting till the episode is finished (Equation 2-15). At time t+1, TD methods form a TD target ($r_t + \gamma \cdot V(s_{t+1})$) and make a useful update, where, $r_t$ denotes the observed reward; $\gamma$ denotes discount factor; $V(s_{t+1})$ denotes for approximated state value function in state $s_{t+1}$.

$$V(s_t) \leftarrow V(s_t) + \alpha(r_t + \gamma V(s_{t+1}) - V(s_t)) \quad \text{(Equation 2-15)}$$

Q-learning is an off-policy TD algorithm. Q-learning uses a lookup table called a Q-table that stores approximated action value functions called Q values, for a specific action with respect to a finite set of state-action pairs. Q-learning updates the action value functions based on the experience sample $<s_t, a_t, r_t, s_{t+1}>$ and the update rule is defined by Equation 2-16. As Q-learning updates the Q-values without executing action ($a_{t+1}$) in the next state, policy exploration is independent of the policy being followed which is advantageous for the development of an optimal policy.

In other words, Q-learning updates action value functions using only one step experience samples in model-free manner. In addition, policy development is achieved through repetitive experiences and self-learning. However, Q-learning is

inappropriate for problems that can have large number of states and actions because of the memory storage and computation time required to update Q-table.

$$\mathrm{Q}(s_t, a_t) \leftarrow \mathrm{Q}(s_t, a_t) + \alpha(r_t + \gamma \underset{a' \in A}{\operatorname{argmax}} Q(s_{t+1}, a') - \mathrm{Q}(s_t, a_t)) \quad \text{(Equation 2-16)}$$

### 2.2.2 Deep Q-learning

Deep Q-learning is a combination of Q-learning and deep neural network. As mentioned above, sometimes it is inefficient or even impossible to have a Q-table because of large state-action pairs, which is called curse of dimensionality. Deep Q-learning solves this problem by approximating the Q-table using deep neural network, called deep Q-network (DQN). Deep Q-learning gathers experience samples, <s, a, r, s'>, which are stored in replay memory at each time-step and develop policies using the samples. Besides, using replay memory, the correlation of time-series data sampled following an incomplete policy is removed. Also, to overcome the limited number of experiences, deep Q-learning randomly extract samples from the replay memory and offers a variety experiences for agents to develop policies.

In a DQN, exploration evaluates possible actions, whereas exploitation uses knowledge from prior experience. The trade-off between exploration and exploitation must be carefully considered. The $\epsilon$-greedy method is one of the most popular techniques for balancing exploration and exploitation, where the agent chooses actions with the knowledge with probability of $1 - \epsilon$ and chooses random action with probability of $\epsilon$ for exploration.

Deep Q-learning uses two networks: main Q-network ($\mathrm{Q}(\mathrm{s}, \mathrm{a}; \theta)$) and target Q-network ($\mathrm{Q}(\mathrm{s}, \mathrm{a}; \theta^{-})$). Main Q-network is the decision maker of DQN to maximize the action value function and the target Q-network is a copied network that creates target values for updating the other network. In other words, the action value function

of the current state-action pair (output #1) is derived from the main Q-network and the maximum action value function of the next time-step (output #2) is derived from the target Q-network. At every iteration i, parameters of the main Q-network are updated to minimize the loss between output #1 and output #2 (Equation 2-17). The reason why deep Q-learning uses two networks is that output of the main Q-network can be unstable and unstable outputs are inappropriate for updating the policies for deep Q-learning.

$$L(\theta) = \{Q(s, a; \theta) - (r + \gamma \underset{a' \epsilon A}{arg\max} Q(s', a'; \theta^{-})\}^2 \quad \text{(Equation 2-17)}$$

The structure is shown in Figure 2: 1) Main Q-network chooses the action that maximizes the action value function and sends to the environment; 2) the state of environment changes due to the action and the reward is calculated; 3) experience sample, <s, a, r, s'>, is stored at replay memory; 4) every i, Q-value of current state-action pair is calculated using main Q-network and the maximum value of future Q-value is calculated using target Q-network; 5) the parameters of main Q-network are updated to minimize the loss function; 6) main Q-network is cloned to target Q-network.

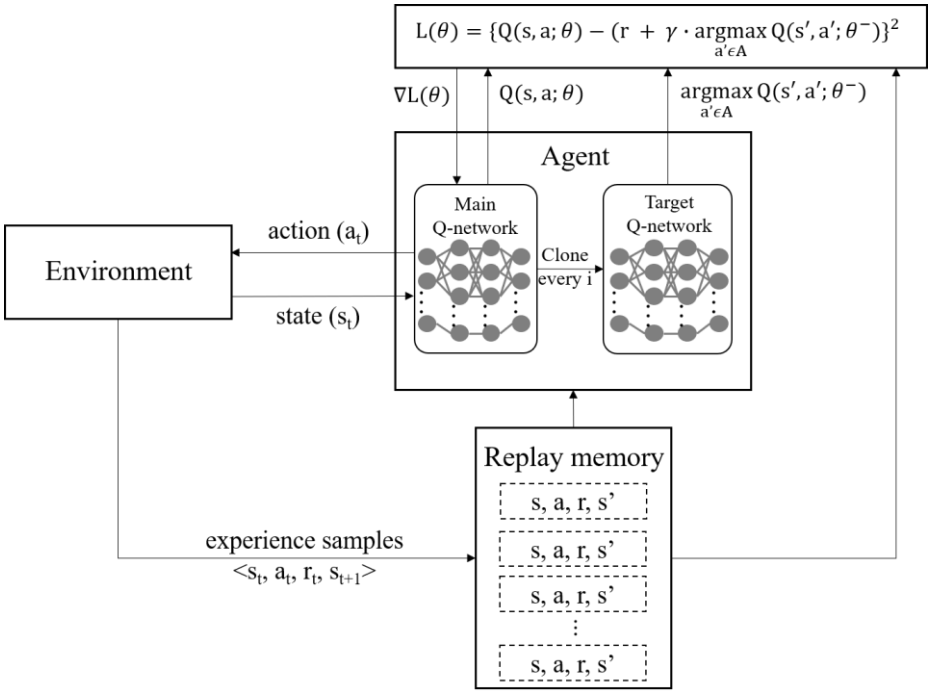



**Figure 2-2. Deep Q-learning (Ahn et. al., 2020)**

## 2.3. Previous works to implement reinforcement learning to existing buildings

Reinforcement learning based control of building systems can be model-free approach, but the RL controllers take unacceptably long time to be trained. Thus, directly implementing such controllers would not be practical in real building control problems. There are very few studies conducted on implementing reinforcement learning algorithms to existing buildings. The studies developed models capable of simulating systems and trained RL agents using the models.

There are several studies that developed physics-based simulation model for RL control. Yang et. al. (2015) applied Q-learning to control the circulation flow rate of the solar system based on the solar radiation and outdoor air temperature in a residential building. The performance of RL controller is compared to rule-based control using Simulink model of the target building and the result shows that 11% of energy cost can be saved by the RL controller. However, model calibration is not conducted to reduce performance gap between the simulation model and real building. This means that the dynamics the RL agent learned may differ from that of the real building, and the performance of the RL controller may not be maintained when applied to the real building. Zhang et. al. (2019) applied reinforcement learning algorithm to control a radiant heating system of an existing office building. A whole building energy model is developed using EnergyPlus to provide pre-training opportunity for the agent to learn dynamics of the building before implemented to real situation. Calibration process is conducted to minimize the error between simulated result and measured data of energy use and indoor air temperature. The result shows that 17% of heating demand is saved by reinforcement learning based control compared to rule-based control. However, for most of existing buildings, developing well-calibrated physics-based models is challenging because of lack of detailed data required for calibration. Also, as there are many kinds of novel HVAC systems that cannot be modeled by current modeling tools, the framework is not generalizable to existing buildings.

To overcome the limitations of physics-based simulation model for RL based control, some studies used data-driven models as system simulation models. Kazmi et. al. (2018) controlled hot water controllers in 32 Dutch houses based on reinforcement algorithm with assumption that the problem is Markov Decision Process (MDP). A data-driven prediction model that predicted hot water temperature and the RL controller explored optimal strategies just based on the single variable. The result showed that the proposed controller can save almost 20% of energy consumption. May et. al. (2019) controlled window operation using Q-learning to control indoor air quality. The result showed that RL controller can improve thermal comfort and indoor air quality by 90% compared to manual occupant control. The aforementioned studies just considered a simple single system of buildings and applied Q-learning or simple MDP concept. However, for large scale buildings, numerous facilities interact with each other and determine the level of heat/cold produced and supplied in order to respond to dynamic changes in the environment. In other words, RL agents should be able to consider complex interactions between plant systems to be applied for optimal control problems.

## 2.4. Conclusion

In this section, the basic theories of reinforcement learning and deep Q-learning is introduced. By using deep neural network, deep Q-learning solves the dimensional problem of conventional Q-learning while maintaining its advantages. Its ability to learn the dynamics and find optimal control strategy of complex systems is suitable for large scale buildings equipped with numerous components. From the review of previous works that tried to implement reinforcement learning to real-world cases, the followings are found:

- Some researches proposed practical framework which is combination of physics-based simulation model and model-free reinforcement algorithms. However, the challenge to develop high fidelity physics-based model for existing buildings are not considered. Therefore, the framework is not generalizable.

- Data-driven models are also used to train reinforcement learning agent practically. However, the studies are conducted only considering control of local systems. In order to study the potential of RL, research on whole building system control needs to be conducted.

- Agents in RL algorithms that combine deep learning and reinforcement learning, such as DQN, are black-box models in the form of neural networks. Therefore, it is difficult to interpret the decision-making process of the agents, which is one of challenges for applying RL based control to real buildings. To the best of the authors' knowledge, no studies have been conducted interpreting the decision-making process of artificial agents.

In order to apply reinforcement learning to real building system, building operators should be able to understand artificial intelligence's decision-making process and build "Trust" for it. To this end, studies are being conducted on how to interpret trained RL agents. In the next chapter, a decision tree, a class of explainable machine learning,

are described, and the concept of interpreting RL agents using the decision tree is introduced.

# 3. Decision Trees

A decision tree is an intuitive and explainable machine learning algorithm. A decision tree can be used to discover features and extract patterns in large databases that are important for discrimination and predictive modeling. A decision tree is constructed by recursively dividing the feature space of the training set. The objective of decision tree is to come up with decision rules that provide an informative and robust hierarchical classification model. This section explains the components of a decision tree with a simple example, and introduces a simple but practical method of interpreting decision-making process using a decision tree.

## 3.1 Summary of decision tree

A decision tree (Figure 3-1) is a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The decision tree models consist of nodes and there are three types of nodes: 1) a root node is the starting point of the decision tree; 2) leaf node shows the discriminated class; 3) decision node/internal node shows the rules for classifying data using features of the data.

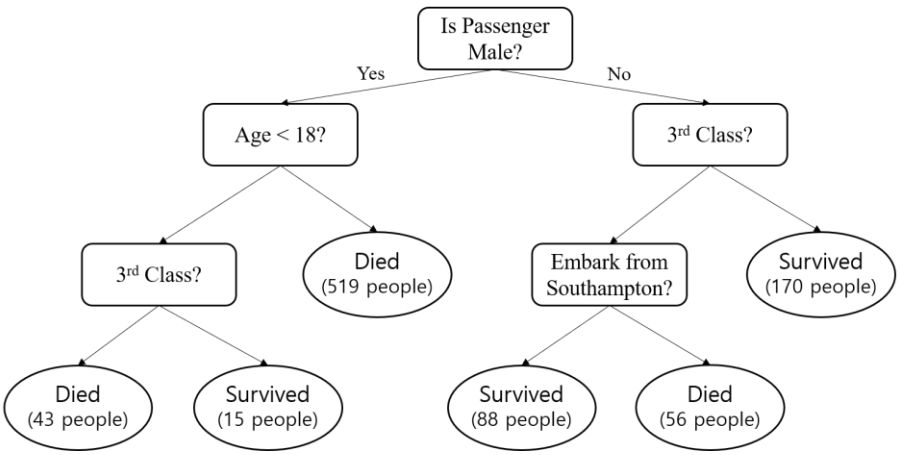



**Figure 3-1. Example of decision tree (Titanic survival prediction)**

In general, a decision tree completes a tree that achieves optimal goals by computing information gain. The information gain is calculated through entropy (Equation 3-1) changes. Entropy represents the degree of disorder and is a numerical expression of the amount of information possessed by the probability distribution. If the probability of a certain value in the probability distribution increases and that of remaining values decreases, the entropy decreases. Information gain refers to the degree to which the entropy decreases by splitting into child nodes. The greater the information gain, the better the classification of data. The decision tree tries various classification methods for one decision node and proceeds with learning in the direction of the largest information gain. In other words, a decision tree finds leaf nodes that can minimize the disorder of data through continuous classification. This process is called recursive partitioning.

$$Entropy(A) = -\sum_{k=1}^{m} p_k \cdot log_2(p_k) \quad \text{(Equation 3-1)}$$

where $m$ denotes the number of classes; and $p_k$ denotes for the ratio of k-class data.

The state in which the decision tree branches over all learning data is called full tree. It is known that a full tree is prone to overfitting for train data, and pruning methods are used to prevent the overfitting. Pruning is stopping generating new nodes if the information gain does not exceed a certain value, which can improve the interpretability of trained decision trees, although classification accuracy may be slightly reduced.

To make pruning efficient, it is necessary to calculate feature importance from full-tree. Feature importance is an index that tells how much the feature in the data affect the exact classification of decision trees. Feature importance is obtained by measuring how different the model prediction results are when the feature is modified. Key features can be selected from feature importance calculations and be used to create interpretable decision trees.

## 3.2 Classification And Regression Trees (CART)

The classification and Regression Trees (CART) algorithm is the most popular decision tree algorithm which is used for both regression and classification. The CART algorithm uses Gini impurity (Equation 3-2) instead of entropy to calculate the information gain. Gini impurity is a measure of how much heterogeneous things are mixed in data. Nodes with 0 Gini impurity are pure. Gini impurity is used as a substitute of entropy because it requires less computation. A decision tree explores rules in the direction of lowering the Gini impurity.

$$\mathrm{I(A)} = 1 - \sum_{k=1}^{m} {p_k}^2 \qquad \text{(Equation 3-2)}$$

where I(A) denotes the gini index of data A; and $p_k$ denotes for the ratio of k-class data.

Applying the CART algorithm, a classification tree is built using "splitting" which is splitting data of parent node to two child nodes that have maximum homogeneity according to splitting rule. As the Gini index of parent node is a fixed value, the maximum homogeneity of child nodes can be calculated by the maximization of change of impurity function. The change of impurity function calculated using Gini index is shown in Equation 3-3.

$$\Delta\mathrm{I(A)} = -\sum_{k=1}^{m} {p_k}^2 + P_1 \sum_{k=1}^{m} {p_{1k}}^2 + P_2 \sum_{k=1}^{m} {p_{2k}}^2 \qquad \text{(Equation 3-3)}$$

where $\Delta\mathrm{I(A)}$ denotes the change of Gini index; *m* denotes the number of classes; $p_k$ denotes for the ratio of *k*-class data; $P_i$ means the probability of being classified to child node *i*; and $p_{ik}$ denotes for the ratio of k-class data in child node *i*.

## 3.3 Interpreting reinforcement learning using decision tree

Because the results of decision trees can be visualized easily, they are used instead of the original black box model for the purpose of interpretation. However, decision trees are not easily learned, so other methods, like distillation, are used to train decision trees in several decision tree-based reinforcement learning. Distillation is guiding a shallow model such as decision trees by a more complex model like deep neural networks. Instead of directly training decision trees, deep neural networks are first trained with the data. Then, the knowledge of deep neural networks is transferred to shallow models by generating soft labeled data.

As mentioned above, reinforcement learning is learning how to map actions to states. In other words, the trained agents of reinforcement learning can generate states data with actions label (Figure 3-2). By training a decision tree using the data, the decision rules of the agents can be interpretable which can be utilized in following two aspects. 1) Verify agents of reinforcement learning: Deriving the rules of actions of the agents can be an indirect way of proving that they can learn the complex dynamics in real world and make decisions accordingly. In other words, it will be possible to increase the reliability of end-users (eg. system operators in buildings) in the reinforcement learning agent and boost the adaptation of reinforcement learning based optimal control for the building systems. 2) Deriving reduced rules: The reliability of end-users in black box networks is not the only reason they are rarely applied in the field. By using decision trees, it is possible to derive reduced rules from the artificial agents which are intuitive and are easily to be applied in the field where physical systems are not yet ready to implement optimal control applying reinforcement learning based optimal control. Reduced rules can be a realistic alternative in building optimal control, which demonstrates how artificial intelligence and humans can collaborate.

In this study, a simple decision tree will be trained using the state-action pairs generated by the deep Q-network. The decision rule of the agent will be derived as a reduced rule (Figure 3-2) that can be directly applied to the target building.

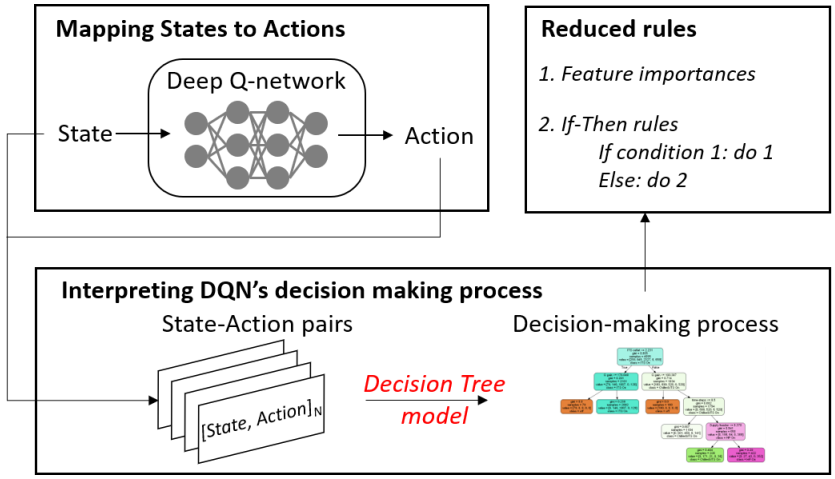
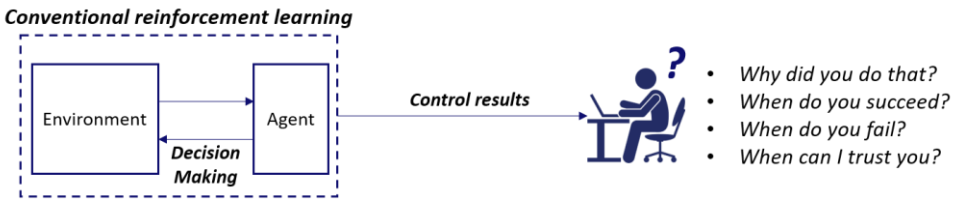


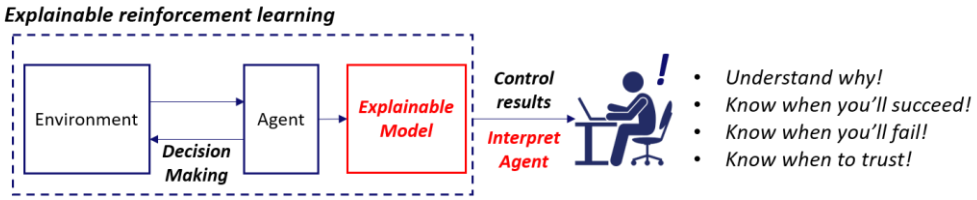

**Figure 3-2. Distill explanation of DQN agent.**

## 3.4 Conclusion

In this section, the basic theories of decision trees are introduced and the concept of distillation explanation of reinforcement learning is introduced. In this study, the explanation for the agent of deep Q-learning will be generated by using a simple decision tree. By interpreting the decision-making process of the artificial intelligence, reliability of the artificial intelligence can be enhanced (Figure 3-3). The next chapter introduces the process of developing simulation models using collected BEMS data to apply explainable reinforcement learning to a real building system.




(a) Conventional reinforcement learning




(b) Explainable reinforcement learning

**Figure 3-3. Conventional vs. Explainable reinforcement learning**

# 4. Target building and Federated model

The project described in this study is conducted to explore the applicability of reinforcement learning based optimal control of an existing building. RL can be model-free approach, but it is risky to directly implement the RL to the building system. Therefore, the model should be developed to explore the applicability and also to use the model to transfer knowledge about the dynamics of building system. In this chapter, a cooling system of the target building is described and initial concept of federated model is proposed as a simulation model developed for pre-training a deep Q-network (DQN). The federated model is an integrated data-driven model that divides a building system into several modules based on physical causality and develops each module into a data-driven model to perform simulations on building systems. The structure of the federated model is introduced and the limitation of the model is also discussed.

## 4.1 Parallel cooling system

The target building is an office building located in Gyeongsangbuk-do, South Korea, with a floor area of 145,864m2, four basement floors, and 28 ground floors. The office zone is divided into lower (5F-16F) and higher (17F-28F) floors. The cooling system of the building consists of ice-based thermal storage system and geo-thermal heat pump system (Figure 4-1). The ITS system consists of two chillers (520 USRT each), an ice-based thermal storage (10,400 USRT), and a heat exchanger (HEX1, Figure 4-1). The geo-thermal heat pump system consists of 18 heat pumps (156 kW each) and a heat exchanger (HEX2, Figure 4-1).

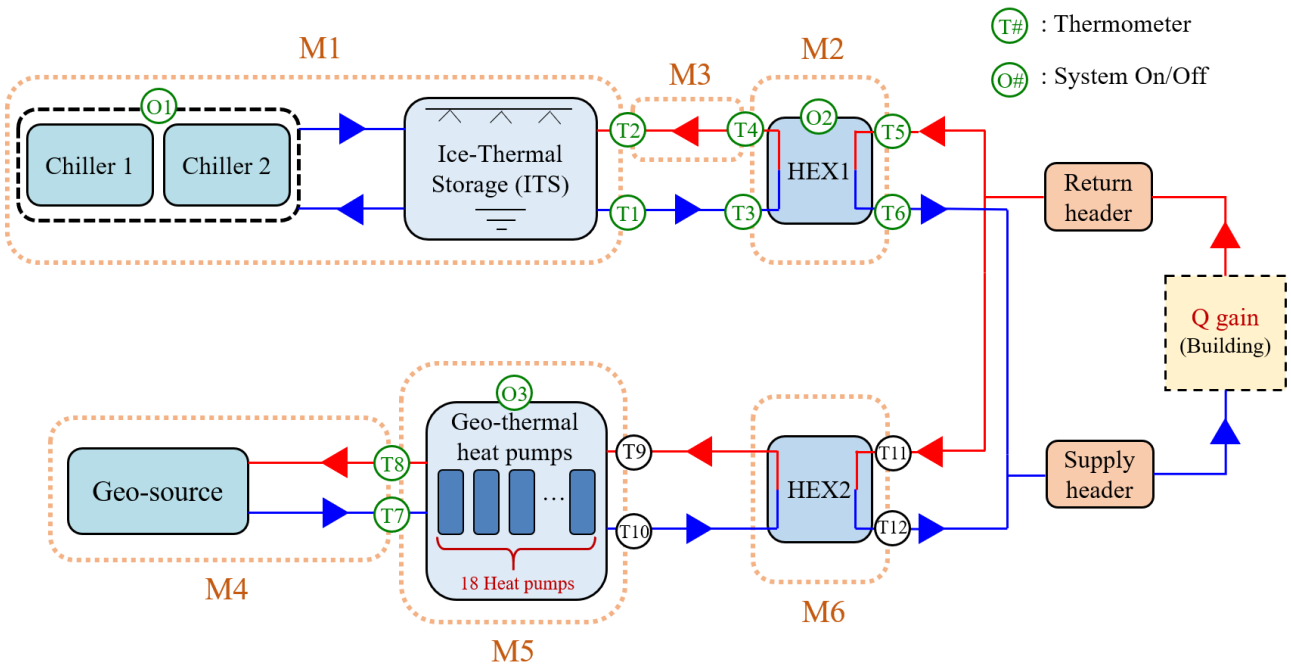



**Figure 4-1. Cooling system (ITS + Geo-thermal HP) of the target building**

To set optimal control objectives and select input and output variables for the system model, data analysis and interviews with building operators are conducted. The target building is collecting an hour interval of data through BEMS and the data gathered during summer season (2017.06.23-2017.08.20) is used for system analysis. Table 4-1 shows a list of the data used in this study and the major results are as followings:

- Data from the ITS is being collected, but the temperature data of geo-thermal heat pump system (T9-T12) is not collected. In addition, the flow rate data of fluid are not collected, and electricity energy use data from the chillers and heat pumps are not separately measured from other devices. Therefore, the flow rates and electric energy use of system components are assumed to be equal to values of nominal specifications.

- Outlet temperature of ITS (T1, Figure 4-1) is controlled not to exceed 3℃ when providing cold to the building. Through interviews with actual building operators, it is found that T1 should be maintained to be lower than 3℃ to properly provide cold for the building.

**Table 4-1. Data gathered through BEMS**

| ID | Measured variables | | Unit |
|---|---|---|---|
| T1 | Ice-thermal storage (ITS) system | ITS outlet temperature | ℃ |
| T2 | | ITS inlet temperature | ℃ |
| T3 | | HEX1 supply side inlet temperature | ℃ |
| T4 | | HEX1 supply side outlet temperature | ℃ |
| T5 | | HEX1 demand side inlet temperature | ℃ |
| T6 | | HEX1 demand side outlet temperature | ℃ |
| O1 | | Chillers On/Off | - |
| O2 | | HEX1 On/Off | - |

**Table 4-1. Data gathered through BEMS (continued)**

| ID | Measured variables | | Unit |
|---|---|---|---|
| T7 | Geo-thermal heat pump system | Geo-thermal pipe outlet temperature | ℃ |
| T8 | | Geo-thermal pipe inlet temperature | ℃ |
| O3 | | Heat pump On/Off | - |
| T9 | | Heat pumps inlet temperature (not measured) | ℃ |
| T10 | | Heat pumps outlet temperature (not measured) | ℃ |
| T11 | | HEX2 demand side inlet temperature (not measured) | ℃ |
| T12 | | HEX2 demand side outlet temperature (not measured) | ℃ |

The ITS system is operated to provide cold for upper office floors of the building (17F-28F) while the heat pump system provided cold for lower office floors (5F-16F), during the target period. Hence the data of the heat pump system is not gathered, this study is conducted on optimal control strategy of the system to provide cold for upper office floors through the parallel cooling system. As control strategies for AHUs are not considered, the aggregated cooling load of the upper office floors is calculated using Equation 4-1.

$$Q = \dot{m} c_p (T5_{t+1} - T6_t) \times (1\ hour) \quad \text{(Equation 4-1)}$$

where Q: aggregated hourly cooling load of the upper office floors (kWh); $\dot{m}$: chilled water flow rate of HEX1 supply side (kg/s); $c_p$: heat capacity of water (kJ/kg·K); $T5_{t+1}$: HEX1 supply side inlet temperature at time-step t+1; $T6_t$: HEX1 supply side outlet temperature at time-step t. Please note that one time-step denotes for an hour.

## 4.2 Federated model

To explore optimal control strategies of the cooling system, a federated model, a novel concept of simulation model, is proposed. Federated model is an integrated data-driven model that divides a building system into several modules based on physical causality and develops each module into a data-driven model to perform simulations on building systems. Each module in the federated model shares input variables and performs sequential simulations using output variables from other modules as input variables. The advantages of federated model are as followings:

Structural flexibility of the simulation model can be enhanced: Conventional building simulation tools, such as EnergyPlus, have pre-defined topology rules which make it hard to develop a simulation model identical to the real building. For example, in EnergyPlus, libraries of ice-based thermal storage do not have multiple outlet nodes but, the ITS of the target building has two outlet nodes as shown in Figure 4-1. Although EnergyPlus provides splitter and mixer libraries, they also have limitations because the inlet and outlet topology rules of them should be matched. By developing federated model, components of the model can be connected identical to the real system which means subjective assumptions of the model developer may be less involved.

Data-driven simulation model can be developed without calibration: When developing physics-based simulation models, model calibration is essential to reduce the performance gap between the simulation results and the real operational data. However, calibration process typically lacks critical inputs from physics and engineering perspectives, thus sometimes leading to unreasonable calibrated results. By directly developing data-driven simulation models using real operational data, high fidelity simulation models that reflect the dynamics of target building can be developed without calibration process.

The system is divided into six modules (M1-M6, Figure 4-1). ITS system is modeled using three modules (M1-M3): M1, containing two chillers and ITS, predicts outlet temperature of ITS (T1). M2 predicts brine outlet temperature of HEX1 (T4) and

chilled water outlet temperature of HEX1 (T6). In other words, M2 calculates the amount of cold provided by the HEX1 when the ITS system is operated. M3 predicts the inlet temperature of ITS (T2). Although the brine outlet temperature of HEX1 (T4) and the inlet temperature of ITS (T2) are expected to be the same, the actual operational data shows that there is a significant difference between the two temperatures for unknown reasons. Therefore, M3 is developed to predict relationship between T2 and T4.

The geo-thermal heat pump system consists of three modules (M4-M6): M4 predicts source side inlet temperature of heat pumps (T7). M5 calculates source side outlet temperature of heat pumps (T8) and supply side outlet temperature of heat pumps (T10). In other words, M5 calculates the amount of the cold generated by the heat pumps. M6 calculates supply side outlet temperature of HEX2 (T9) and demand side outlet side outlet temperature of HEX2 (T12) which means M6 calculates the amount of heat exchanged through HEX2.

M1-M4 are developed as data-driven model using artificial neural networks (ANNs). ANNs, based on a multi-layer perceptron, have been widely used to resolve engineering problems. ANNs consist of input, output, and hidden layers, and each layer consists of several nodes. The relationship between the nodes is determined by weights and the weights are updated to minimize errors, defined as difference between the output of the model and measurements, through back propagation. ANNs are able to predict and represent the dynamics of buildings systems with high fidelity, based on their ability to reflect non-linearity. Table 4-2 to 4-5 show the inputs and outputs of ANNs. The input variables of each ANN are selected based on domain knowledge and to predict or calculate outputs using historical data collected over a period of time to reflect the effect of the thermal inertia of buildings and systems. The modules are trained using data gathered during 2017.06.23-2017.08.11.

**Table 4-2. Inputs and outputs of M1**

| Variables | Variable type | Unit |
|---|---|---|
| ITS outlet temperature (T1, t-1) | Inputs | ℃ |
| ITS outlet temperature (T1, t-2) | | ℃ |
| ITS outlet temperature (T1, t-3) | | ℃ |
| ITS inlet temperature (T2, t-1) | | ℃ |
| ITS inlet temperature (T2, t-2) | | ℃ |
| ITS inlet temperature (T2, t-3) (output of M3) | | ℃ |
| Chillers On/Off (O1, t) | | - |
| Chillers On/Off (O1, t-1) | | - |
| Chillers On/Off (O1, t-2) | | - |
| ITS outlet temperature (T1, t) | Outputs | ℃ |

**Table 4-3. Inputs and outputs of M2**

| Variables | Variable type | Unit |
|---|---|---|
| HEX1 supply side inlet temperature (T3, t) (output of M1) | Inputs | ℃ |
| HEX1 supply side inlet temperature (T3, t-1) | | ℃ |
| HEX1 supply side inlet temperature (T3, t-2) | | ℃ |
| HEX1 demand side inlet temperature (T5, t) | | ℃ |
| HEX1 demand side inlet temperature (T5, t-1) | | ℃ |
| HEX1 demand side inlet temperature (T5, t-2) | | ℃ |
| HEX1 On/Off (O2, t) | | - |
| HEX1 On/Off (O2, t-1) | | - |
| HEX1 On/Off (O2, t-2) | | - |
| HEX1 supply side outlet temperature (T4, t) | Outputs | ℃ |
| HEX1 demand side outlet temperature (T6, t) | | ℃ |

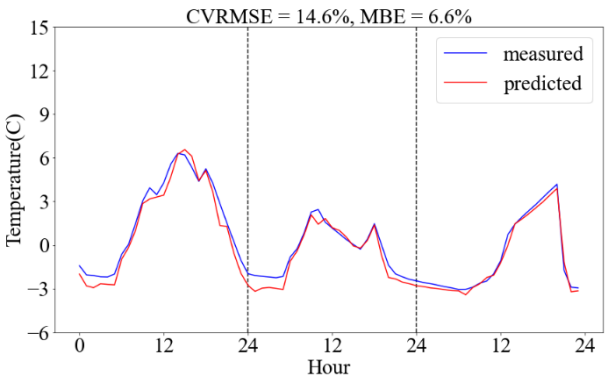
**Table 4-4. Inputs and outputs of M3**

| Variables | Variable type | Unit |
|---|---|---|
| ITS inlet temperature (T2, t-1) | Inputs | ℃ |
| ITS inlet temperature (T2, t-2) | | ℃ |
| ITS inlet temperature (T2, t-3) | | ℃ |
| HEX1 supply side outlet temperature (T4, t) (output of M2) | | ℃ |
| HEX1 supply side outlet temperature (T4, t-1) | | ℃ |
| HEX1 supply side outlet temperature (T4, t-2) | | ℃ |
| Chillers On/Off (O1, t) | | - |
| Chillers On/Off (O1, t-1) | | - |
| Chillers On/Off (O1, t-2) | | - |
| HEX1 On/Off (O2, t) | | - |
| HEX1 On/Off (O2, t-1) | | - |
| HEX1 On/Off (O2, t-2) | | - |
| ITS inlet temperature (T2, t) | Outputs | ℃ |

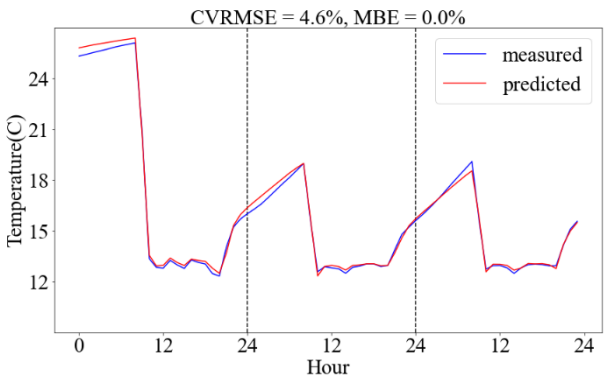
**Table 4-5. Inputs and outputs of M4**

| Variables | Variable type | Unit |
|---|---|---|
| Geo-thermal pipe outlet temperature (T7, t-1) | Inputs | ℃ |
| Geo-thermal pipe outlet temperature (T7, t-2) | | ℃ |
| Geo-thermal pipe outlet temperature (T7, t-3) | | ℃ |
| Geo-thermal pipe inlet temperature (T8, t) (output of M5) | | ℃ |
| Geo-thermal pipe inlet temperature (T8, t-1) | | ℃ |
| Geo-thermal pipe inlet temperature (T8, t-2) | | ℃ |
| Heat pump On/Off (O3, t) | | - |
| Heat pump On/Off (O3, t-1) | | - |
| Heat pump On/Off (O3, t-2) | | - |
| Geo-thermal pipe outlet temperature (T7, t) | Outputs | ℃ |

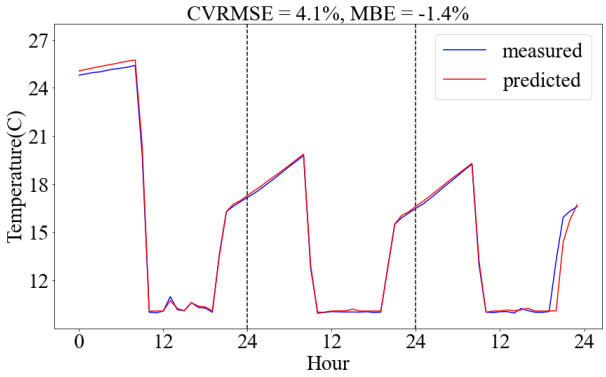
Figure 4-2 shows the validation results of data-driven modules. The validation is conducted using data gathered for 3 days (2017.08.16-2017.08.18) which are not used to train the models. As a result of the validation, the prediction accuracy for the five output variables from each module satisfies the hourly prediction accuracy (less than CVRMSE 30% and less than MBE 10%) recommended by ASHRAE Guideline 14 (2002).



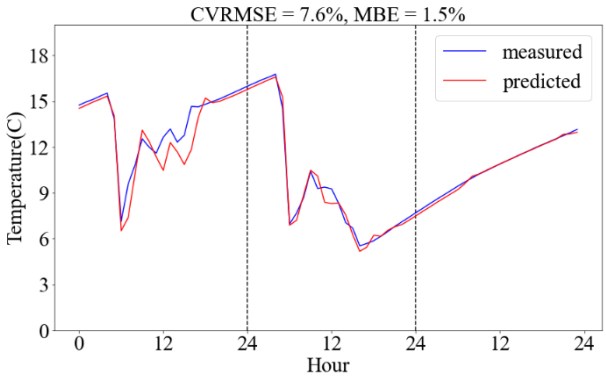

(a) Model accuracy of M1 (prediction of T1)



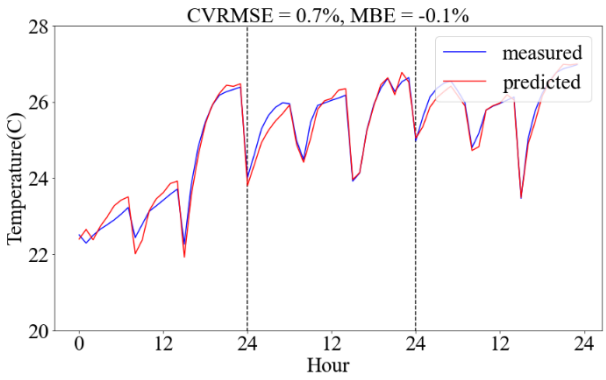

(b) Model accuracy of M2 (prediction of T4)




(c) Model accuracy of M2 (prediction of T6)




(d) Model accuracy of M3 (prediction of T2)




(e) Model accuracy of M4 (prediction of T7)

**Figure 4-2. Model accuracy of modules (M1-M4)**

M5 and M6 are developed using physics-based equations because the operational data of heat pumps and HEX2 (T9-T12) are missing. M5 consists of two equations: calculating cold supplied by heat and calculating electric energy use of the heat pumps (Equation 4-2 - 4-3, DOE, 2019a).

$$\frac{Q_c}{Q_{c,ref}} = A1 + A2\left[\frac{T_{L,in}}{T_{L,ref}}\right] + A3\left[\frac{T_{S,in}}{T_{S,ref}}\right] + A4\left[\frac{\dot{V_L}}{\dot{V_{L,ref}}}\right] + A5\left[\frac{\dot{V_S}}{\dot{V_{S,ref}}}\right] \quad \text{(Equation 4-2)}$$

$$\frac{Power_c}{Power_{c,ref}} = B1 + B2\left[\frac{T_{L,in}}{T_{L,ref}}\right] + B3\left[\frac{T_{S,in}}{T_{S,ref}}\right] + B4\left[\frac{\dot{V_L}}{\dot{V_{L,ref}}}\right] + B5\left[\frac{\dot{V_S}}{\dot{V_{S,ref}}}\right] \quad \text{(Equation 4-3)}$$

where $Q_c$: supplied cold (W); $Q_{c,ref}$: nominal capacity (W); $T_{L,in}$: demand side inlet temperature (°C); $T_{L,ref}$: nominal demand side inlet temperature (°C); $T_{S,in}$: supply side inlet temperature (°C); $T_{S,ref}$: nominal supply side inlet temperature (°C); $\dot{V_L}$: demand side flow rate ($m^3/s$); $\dot{V_{L,ref}}$: nominal demand side flow rate ($m^3/s$); $\dot{V_S}$: supply side flow rate ($m^3/s$); $\dot{V_{S,ref}}$: nominal supply side flow rate ($m^3/s$); $Power_c$: electric energy use (W); $Power_{c,ref}$: nominal electric energy use (W).

M6 is a module that calculates the amount of heat transferred through the HEX2 and the Number of Transfer Units (NTU) method is used, in this study. NTU method is a calculation method used when there is insufficient information to calculate the Log-Mean Temperature Difference (LMTD). In this study, NTU method is used because only inlet temperatures are available. M6 consists of six equations as follows (Equation 4-4 - 4-9, DOE, 2019a):

$$R_c = \frac{(\dot{m}c_p)_{Min}}{(\dot{m}c_p)_{Max}} \quad \text{(Equation 4-4)}$$

$$\text{NTU} = \frac{UA}{(\dot{m}c_p)_{Min}} \quad \text{(Equation 4-5)}$$

$$\epsilon = \frac{1-\exp[-NTU(1-R_c)]}{1-R_c\exp[-NTU(1-R_c)]} \quad \text{(Equation 4-6)}$$

$$\dot{Q} = \epsilon\left(\dot{m}c_p\right)_{Min}(T_{SupLoop,In} - T_{DmdLoop,In}) \quad \text{(Equation 4-7)}$$

$$T_{SupLoop,Out} = T_{SupLoop,In} - \frac{\dot{Q}}{(\dot{m}c_p)_{SupLoop}} \quad \text{(Equation 4-8)}$$

$$T_{DmdLoop,Out} = T_{DmdLoop,In} - \frac{\dot{Q}}{(\dot{m}c_p)_{DmdLoop}} \quad \text{(Equation 4-9)}$$

where $R_c$: capacity ratio; $(\dot{m}c_p)_{Min}$: minimum value between capacity of supply side and demand side (W/K); $(\dot{m}c_p)_{Max}$: maximum value between capacity of supply side and demand side (W/K); NTU: Number of Transfer Unit; $UA$: capacity of heat exchanger (W/K); $\epsilon$: effectiveness of heat transfer; $\dot{Q}$: heat transfer rate (W); $T_{SupLoop,In}$: inlet temperature of supply side; $T_{SupLoop,Out}$: outlet temperature of supply side; $T_{DmdLoop,In}$: inlet temperature of demand side; $T_{DmdLoop,Out}$: outlet temperature of demand side.

Figure 4-3 shows the overall structure of the federated model. The simulation sequence of ITS system model is M1→M2→M3→M1 and the sequence of geo-thermal heat pump system model is M4→M5→M6→M4. As mentioned above, output variables of each module are used as input variables of the other modules. For example, the output variable of M1 (T1) is used as one of the input variables of M2 (T3, T1 and T3 are same value). The cold is provided to the building through the output variables of heat exchanger modules (T6 of M2 and T12 of M6) and the demand side inlet temperatures of heat exchangers (T5 and T11) are calculated using cooling load of the office building.

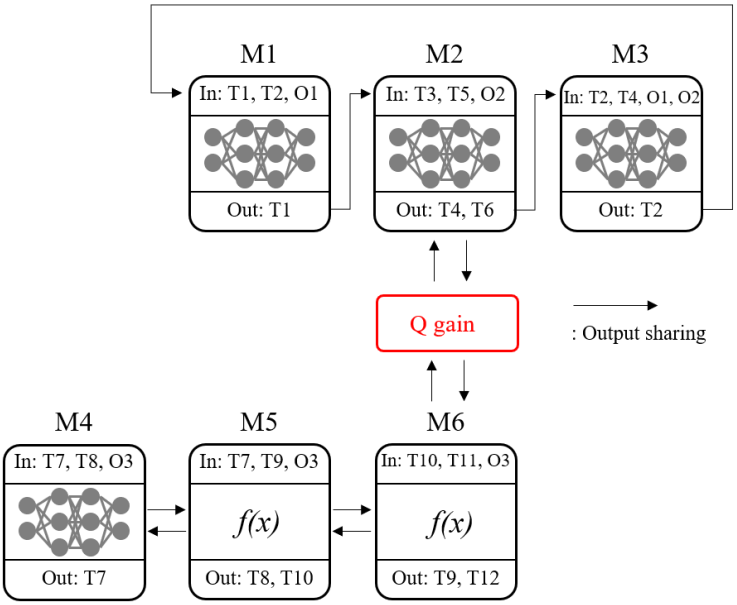



**Figure 4-3. Federated model of the target building**

# 5. Explainable deep Q-network and rule reduction for building control

In Chapter 5, an overall framework of rule reduction for building control using explainable deep Q-network is shown. First, elements (states, actions, and rewards) of reinforcement learning is defined for the implementation of DQN. Especially, control conditions that are required by the building operators are used to define reward functions. Then, control results performed by the DQN agent is compared to the baseline control. Finally, explanation of the DQN agent is shown and reduced rule is extracted from the explanation. The control performance of the reduced rule is also compared to the baseline control and optimal control performed by the DQN agent. It is shown that reduced rule can be practical approach of implementing RL based optimal control to building system.

## 5.1 DQN implementation framework

To implement Deep Q-network (DQN) for optimal control of the target building, states, actions, and reward function is defined based on the data analysis result described in Chapter 4. Table 5-1 shows the states. Temperatures of the system nodes (s1-s12), operating information of chillers, heat pumps, and heat exchangers (s13-s15), electric energy use of the system components (s16-s17), cooling load of the building (s18), and time-flag (s19) are used as states. Please note that time-flag denotes for the hour of the day because the simulations in this study are conducted at an hour interval. Also, the electric energy use of the system components is calculated using nominal specifications because the information is not measured properly.

**Table 5-1. States defined for pre-training of DQN agent**

| ID | State | Unit |
|---|---|---|
| $s_1$ | ITS outlet temperature | ℃ |
| $s_2$ | ITS inlet temperature | ℃ |
| $s_3$ | HEX1 supply side inlet temperature | ℃ |
| $s_4$ | HEX1 supply side outlet temperature | ℃ |
| $s_5$ | HEX1 demand side inlet temperature | ℃ |
| $s_6$ | HEX1 demand side outlet temperature | ℃ |
| $s_7$ | Geo-thermal pipe outlet temperature | ℃ |
| $s_8$ | Geo-thermal pipe inlet temperature | ℃ |
| $s_9$ | Heat pumps inlet temperature | ℃ |
| $s_{10}$ | Heat pumps outlet temperature | ℃ |
| $s_{11}$ | HEX2 demand side inlet temperature | ℃ |
| $s_{12}$ | HEX2 demand side outlet temperature | ℃ |
| $s_{13}$ | Chillers On/Off | - |
| $s_{14}$ | Heat pumps On/Off (same as HEX2 On/Off) | - |
| $s_{15}$ | HEX1 On/Off | - |
| $s_{16}$ | Electric energy use of chillers | kWh |
| $s_{17}$ | Electric energy use of heat pumps | kWh |
| $s_{18}$ | Cooling load of upper office floors | kWh |
| $s_{19}$ | Time-flag (7-20) | - |

Table 5-2 shows the actions that the agent could take at each state. The first action is to provide cold to the building using ITS without running the chillers. The second action is to provide cold using ITS with running the chillers. When providing cold using the ITS system and if the agent determines that the cooling capacity of the ITS is depleted, the agent selects a2. a3 denotes for the action that operates geo-thermal heat pumps to provide cold without running chillers to charge the ITS. a4 denotes for the action that operates geo-thermal heat pumps while running chillers to charge the

ITS. Although it is disadvantageous to operate chillers for immediate rewards, a4 can be chosen if it is advantageous for future states and delayed rewards.

**Table 5-2. Actions defined for pre-training DQN agent**

| Action | Chillers | Heat pump | Heat Exchanger |
|---|---|---|---|
| $a_1$ | ON | OFF | HEX1 |
| $a_2$ | OFF | OFF | HEX1 |
| $a_3$ | ON | ON | HEX2 |
| $a_4$ | OFF | ON | HEX2 |

As explained in Chapter 2, an agent of DQN explores control strategies that maximizes return, the accumulated reward. Equation 15 shows a reward function defined to evaluate the actions taken by the agent. The goal of the DQN agent in this study is to remove the cooling load, minimize energy consumption, and maintain the system node temperature in line with the building operators' requirements. Therefore, the reward function (Equation 5-1) consists of three main terms: (1) Removal penalty of cooling load (Equation 5-2) represents the difference between the cold provided by the system and the cooling load of the building. When the amount of the provided cold is bigger than the cooling load, the penalty is defined as zero. (2) Energy consumption (Equation 5-3) is calculated as the sum of chillers energy use and heat pumps energy use. (3) Failure penalty is defined for two cases. First, when providing cold using ITS even though outlet temperature of ITS is over 3℃, the failure penalty is obtained because the cooling capacity of the ITS can be determined to be exhausted if the outlet temperature of ITS is over 3℃. Also, as the temperature of supply header must be less than 12℃ for efficient heat exchange to occur in AHUs, the failure penalty is obtained if the temperature of the supply header is higher than 12℃. To sum up, the reward function is proposed to train the agent to explore the control strategies that can provide cold to the building while minimizing energy consumption

of the system and avoiding system failure status. To make the reward a positive value, constant numbers a, b, c, and the failure penalty is set by the trial-error manner: a= 1.2, b = 1.4, c = 1.0.

$$r(t) = \begin{cases} a - b \cdot P_{remove}(t) - c \cdot E(t) & (if\ system\ fail = False) \\ P_{failure}(t) & (if\ system\ fail = True) \end{cases} \quad \text{(Equation 5-1)}$$

$$P_{remove}(t) = \begin{cases} \left|Q_{gain}(t) - Q_{remove}(t)\right| & (if\ Q_{gain}(t) > Q_{remove}(t)) \\ 0 & (if\ Q_{gain}(t) < Q_{remove}(t)) \end{cases} \quad \text{(Equation 5-2)}$$

$$E(t) = s_{16}(t) + s_{17}(t) \quad \text{(Equation 5-3)}$$

where $P_{remove}(t)$ denotes for removal penalty that is defined as the difference between cooling load and cold provided by the system at time-step t; $E(t)$ represents total energy use of the system at time-step t; $P_{failure}(t)$ denotes for failure penalty that the agent gets as reward when the action chosen by the agent make the system to be failed at time-step t; $Q_{gain}(t)$ represents cooling load at time-step t; and $Q_{remove}(t)$ represents cold provided at time-step t.

As balancing between exploitation and exploration is critical for training the agent of reinforcement learning, e-greedy algorithm is used and the e value is defined as Equation 5-4. The e value decreased as the episode of training process increased which means the agent tried to find new control strategies in the early training episodes and after sufficient training, the agent controlled the system using the knowledge learned from exploration.

$$\mathrm{e} = 1/[1 + \exp\left(-5 + \frac{\mathrm{N(episodes)}}{20}\right)] \quad \text{(Equation 5-4)}$$

where e denotes for e value; and N(episodes) represents for the $n^{th}$ episode.

Figure 5-1 shows the framework of pre-training a deep Q-learning agent using the federated model. The pre-training of the artificial agent is conducted using the data gathered for 15 days (2017.07.22.-2017.08.11.) and the agent is trained for 500 episodes to explore optimal control strategies and learn the dynamics of the system.

The federated model and DQN algorithm are realized using python keras module. DQN consist of 3 hidden layers that had 20 nodes each. The size of the replay memory is set as 10,000 and deleted the oldest experience data. The training of DQN is conducted at the end of each episode and the cloning weights of networks is also performed at the end of each episode.

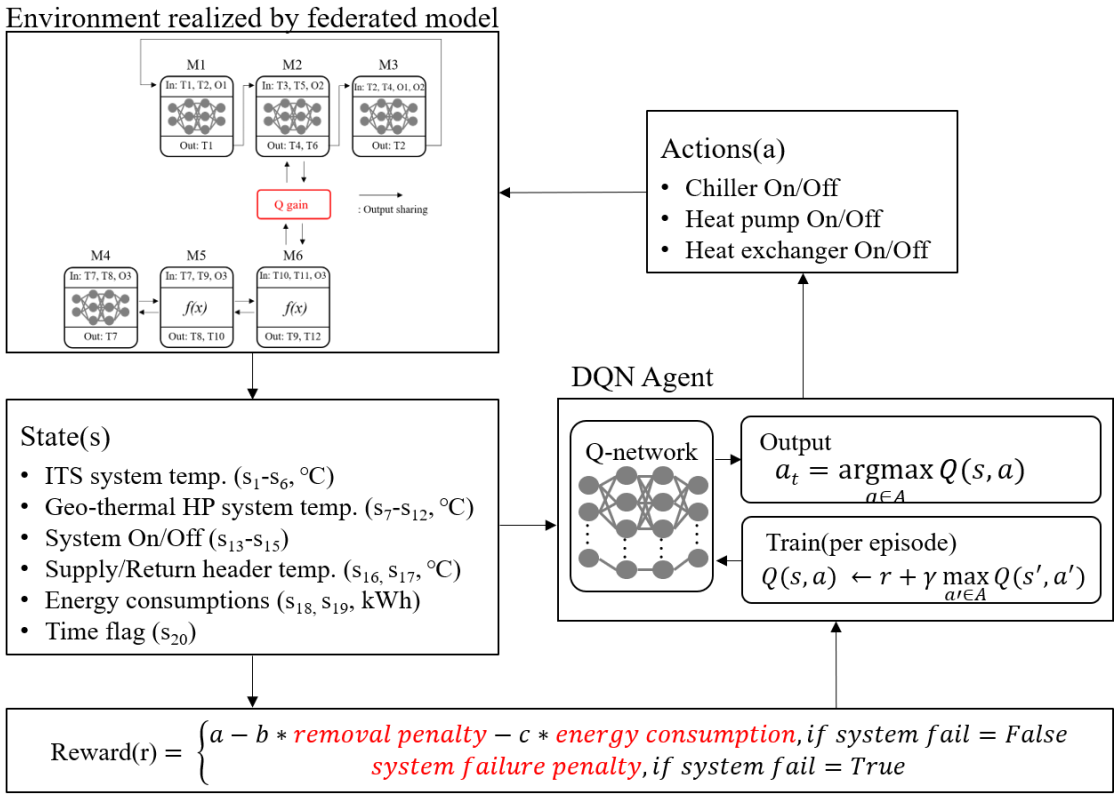
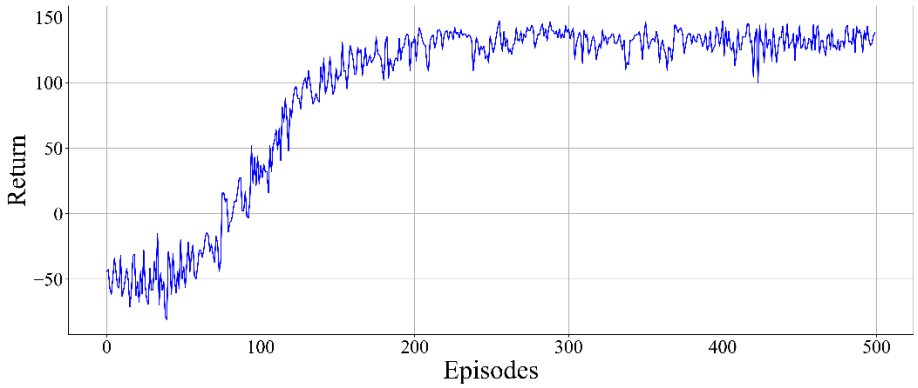


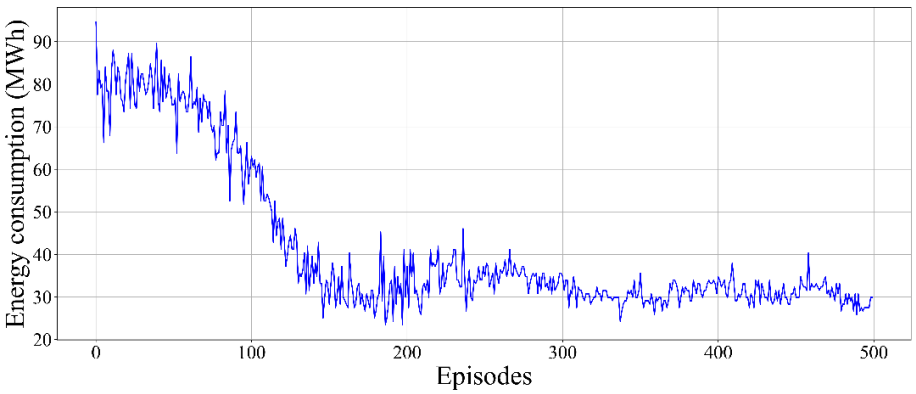

**Figure 5-1. Pre-training framework of DQN agent using the federated model**

## 5.2 Control results of DQN

The DQN agent of this study explores optimal control strategies for daytime operation of the parallel cooling system and the nighttime operation is assumed to be same as the baseline operation. Figure 5-2 shows the return and total energy consumption of the system during training period (July 22 - August 11) per episode. The agent is trained to maximize the return value while minimizing energy consumption of the system according to the definition of the reward function described above. Although the strategy of agent converged after 200 episodes and return value converged to about 142, the agent's control strategy can have uncertainty because the agent continued to update control strategies by self-learning using replay memory (Figure 5-2(a)).




(a) Return per episode during training period




(b) Energy consumption of the cooling system per episode

**Figure 5-2. Training result of the DQN agent (July 22 – August 11)**

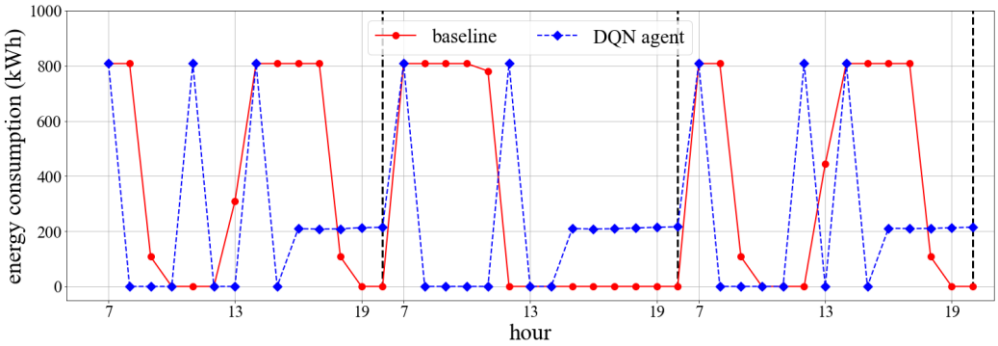
Since it is hard to directly apply the DQN controller to the target building, performance of the controller is validated for 3 days (2017.08.12.-2017.08.14., Table 5-3) using federated model. For validation, DQN agent of 450th episode is used, which derives the return value at most (144.56) during the training process. Table 1 shows the performance of the trained DQN agent compared to the baseline control. DQN agent could save 31.9% of daytime electric energy consumption of the cooling system compared to the baseline control (9,836 kWh vs. 14,440 kWh). This is because the electric energy consumption of daytime chiller operation is reduced by 46% while the agent used heat pumps more than the baseline control. This means that the agent found energy use of the system could be saved by balancing between two systems: ice thermal storage system and geo-thermal heat pump system.

**Table 5-3. Comparison of control performance: Baseline vs. DQN**

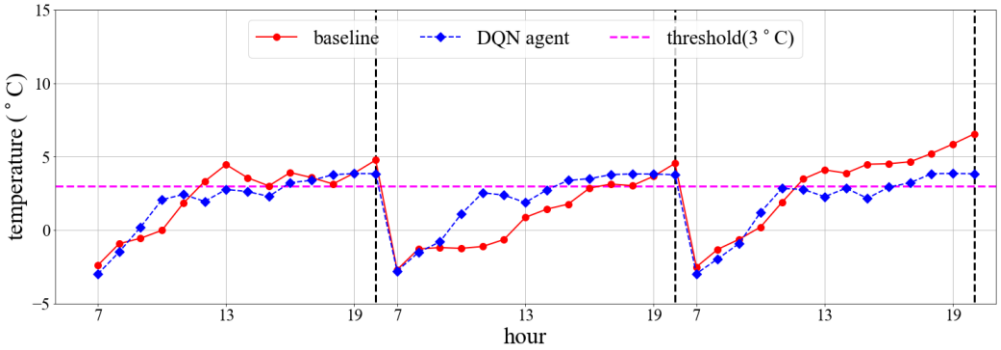
| | **Baseline** | **DQN** |
|---|---|---|
| **Chiller energy consumption (kWh)** | 14,440 | 6,646 |
| **Heat pump energy consumption (kWh)** | 0 | 3,372 |
| **Total energy consumption (kWh)** | 14,440 | 9,836 |
| **System failure hours (out of 42)** | 18 | 6 |
| **Return** | 3.46 | 27.56 |

Figure 5-3 also depicts the performance of the DQN agent compared to the baseline control. The daytime operating duration is 14 hours (07:00 A.M.-20:00 P.M.), so the control performance of the agent is verified for total 42hours over three days. Figure 5-3 (a) shows hourly total energy consumption of the cooling system. For the daytime operation, only chillers are used by the baseline control but the DQN controller operated heat pumps during late afternoon (usually after 16:00 P.M.) when the cooling load is reduced and saved energy consumption of the system. Figure 5-3 (b) shows outlet temperature of ice-based thermal storage (ITS). As described in Chapter 4, the system operators required the outlet temperature of ITS to be below 3℃. For

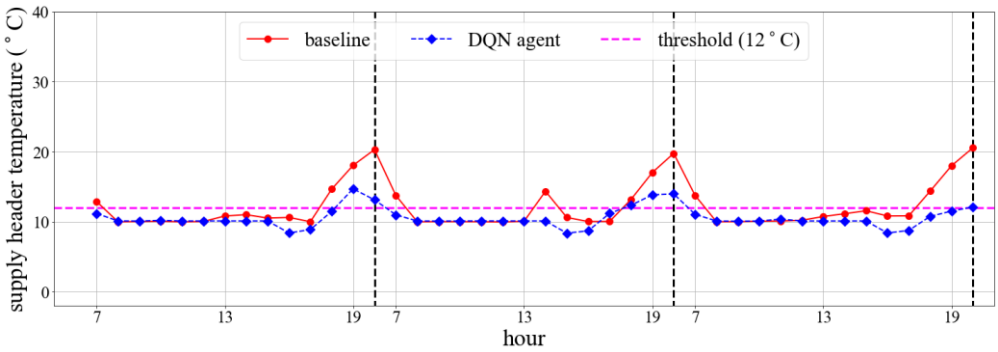
validation period, the baseline control could not meet the requirements for 18 hours out of 42 hours while the DQN agent violated the requirements only for 6 hours. Figure 5-3 (c) shows the supply header temperature of the cooling system. As mentioned above, the artificial agent should find the control strategy that maintains the supply header temperature below 12℃ for proper cold supply. The baseline control could not maintain the temperature below 12℃ for 13 hours during the validation period. However, control by the DQN agent allowed the temperature to exceed 12℃ for only 3 hours. The overall results show that the DQN agent could explore the control strategy that can meet the operational requirements while reducing total energy consumption of the cooling system.




(a) Energy consumption of the cooling system




(b) Outlet temperature of ice-based thermal storage




(c) Temperature of supply header

**Figure 5-3. Control performance: Baseline vs. DQN (August 12-14)**

## 5.3 Rule reduction from DQN agent

During episodes 450–500, 9,750 state-action pairs are generated by the DQN agent as the result of control. As described above, the number of states that the agent used for decision making is 19 and the agent chose one control action for each time-step. Although full decision making rules could be derived using the state-action pairs, the tree is too complicated and inappropriate for interpreting the decision making process, as the full tree consisted of too many leaf nodes and decision nodes.

To make the decision tree interpretable and to perform rule reduction, feature importance of the states is calculated from the full tree. Figure 5-4 shows the feature importance of the states for decision making process. States of which the feature importance is over 0.05 are selected as important features and the selected five important features are: (1) outlet temperature of ice thermal storage ($s_1$); (2) hourly cooling load of the target building ($s_{18}$); (3) demand side outlet temperature of heat exchanger 2 ($s_{12}$); (4) time-flag ($s_{19}$); (5) brine inlet temperature of heat exchanger 1 ($s_3$). Among the five features, as $s_1$ and $s_3$ are basically the same data, only four features ($s_1$, $s_{12}$, $s_{18}$, $s_{19}$) are used to explain decision making process of the DQN agent.

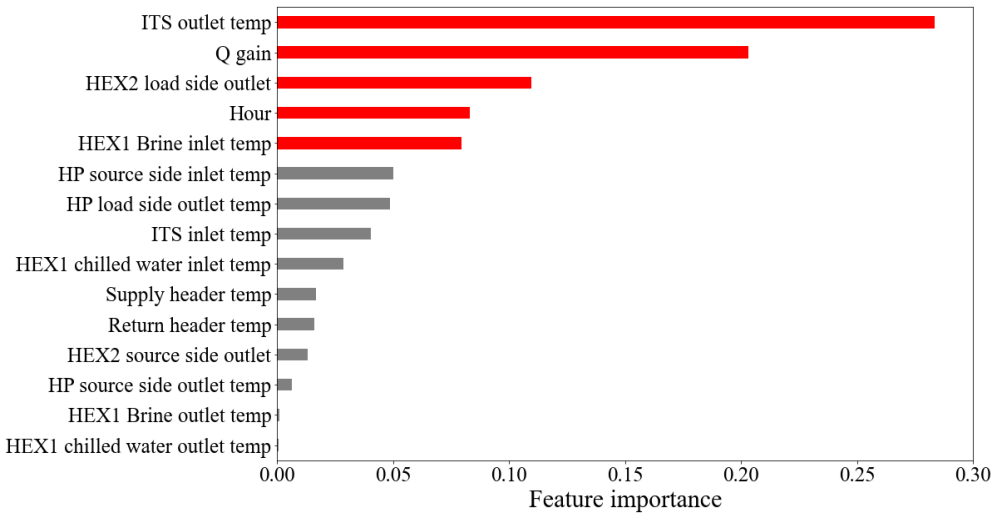



**Figure 5-4. Feature importance for decision making process of the agent**

Figure 5-5 depicts the interpretation of decision making process of the DQN agent

derived using decision tree. To make the decision tree practical, maximum depth of the tree is set as four, minimum impurity decrease of the nodes is 0.02, and minimum samples to split is set as 100. The decision tree shows how the control of DQN agent is performed. For example, if the outlet temperature of ice thermal storage is 2.2℃ or less and hourly cooling load is above 160kWh, the agent used stored ice to provide cold without using chillers. The reference outlet temperature of ice thermal storage is 2.2℃, lower than the specific control condition required by the building operators (3℃), because the agent of reinforcement learning established a control strategy that took into account the future states of the system and building as well as current states.

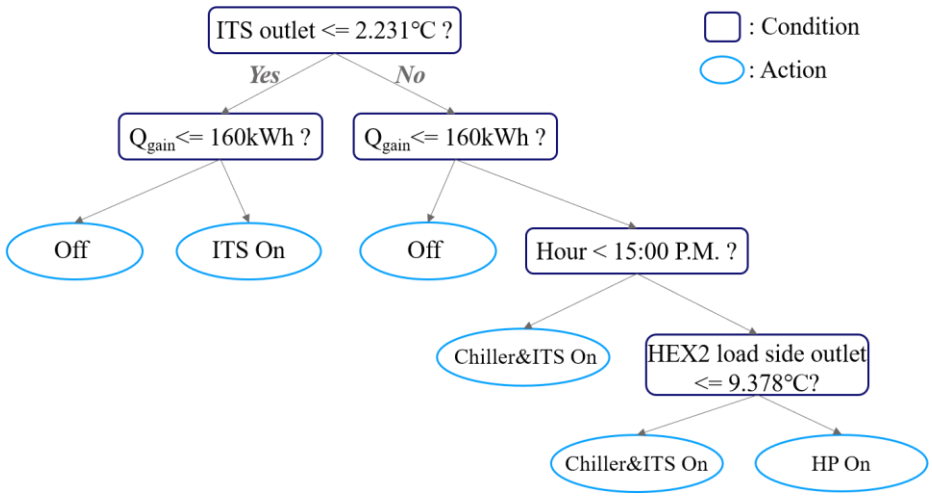



**Figure 5-5. Reduced rule extracted using decision tree**

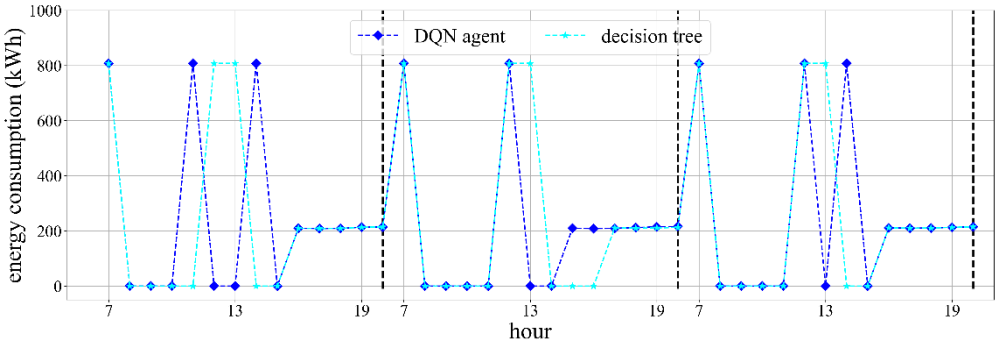
As above, simple “If-then” rules can be derived by tracking nodes and branches of the decision tree and the rules is applied for building control as reduced rules. Table 5-4 and Figure 5-6 show the difference between in performance of reduced rule-based control and the DQN controller for the validation period (2017.08.12.-2017.08.14.). The DQN controller could save more energy compared to the decision tree controller. (9,836 kWh vs. 10,229 kWh). Also, the system failure hours are less caused by DQN controller compared to the reduced rule-based controller (6 hours vs. 7 hours).

However, the reduced rule-based controller could reduce daytime energy consumption by 29.1% (10,229 kWh vs. 14,440 kWh) and also reduce the system failure hours (7 hours vs. 18 hours).

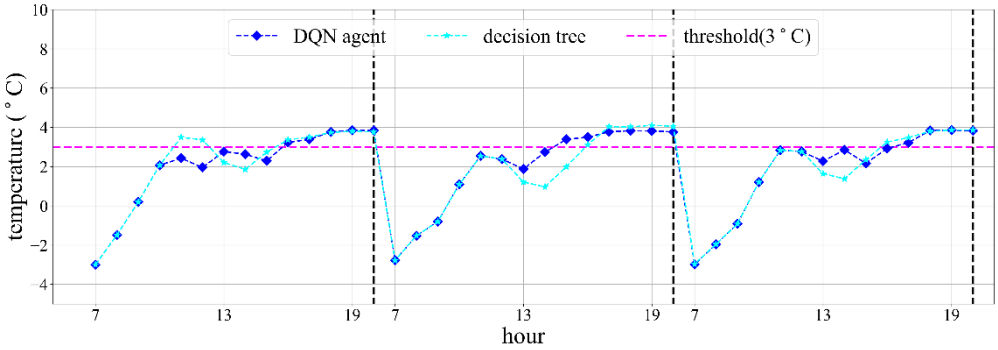
**Table 5-4. Comparison of control performance: DQN vs. Reduced rule**

| | **DQN** | **Decision Tree** |
|---|---|---|
| **Chiller energy consumption (kWh)** | 6,646 | 7,272 |
| **Heat pump energy consumption (kWh)** | 3,372 | 2,957 |
| **Total energy consumption (kWh)** | 9,836 | 10,229 |
| **System failure hours (out of 42)** | 6 | 7 |
| **Return** | 27.56 | 23.17 |

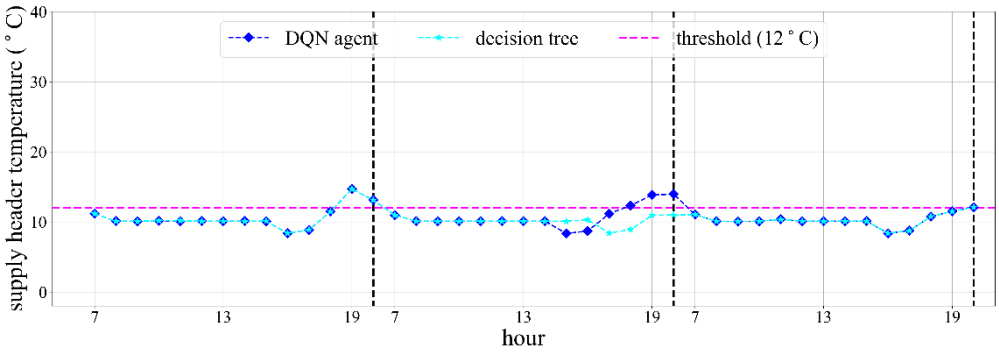
Although the rule is derived using state-action pairs that the DQN controller generated, the control strategies of two controllers (DQN and reduced rule-based) are different (Figure 5-6(a)). This is because the rule is derived using the decision tree generated only using four important features. DQN controller could maintain the outlet temperature of the ice thermal storage below 3℃ for entire hours when using ITS system to provide cold while the reduced rule-based controller violated the condition for 3 hours during the period (Figure 5-6(b)). However, the temperature of supply header is better maintained by the rule-based controller compared to the DQN controller (3 hours vs. 6 hours, Figure 5-6(c)). The overall control performance comparison shows that the simple but control based on rules that are evaluated quantitatively can be practical alternative of complicated optimal controllers.




(a) Energy consumption of the cooling system




(b) Outlet temperature of ice-based thermal storage




(c) Temperature of supply header

**Figure 5-6. Control performance: DQN vs. Reduced rule (August 12-14)**

## 5.4 Discussion

The result of DQN control showed that through iterative learning, the DQN agent could learn dynamics of the target building and the cooling system, and find the balance between different components of the system to reduce energy consumption. To interpret decision making process of DQN agent and extract reduced rule for building control, decision tree is trained using state-action pairs generated by the agent during training period. The control performance of reduced rule using a decision tree is compared to the performance of baseline control and control of DQN agent. The reduced-rule based control is proved to be good-enough compared to complex control of artificial intelligence and the difference in energy savings between the two is marginal, resulting in 2.8%. In other words, it is shown that rules with quantitative evaluation can perform as reduced rule-based control which is practical and reliable way to realize collaboration of human and artificial intelligence.

However, there are still several limitations that the interpretation of the DQN agent's the decision making process using decision tree could not solve and one of the biggest questions remained is: Why the agent judged the four states as major features? The agent judged (1) outlet temperature of ice-based thermal storage ($s_1$); (2) hourly cooling load of the target building ($s_{18}$); (3) demand side outlet temperature of heat exchanger 2 ($s_{12}$); (4) time-flag ($s_{19}$) as four major states for its decision making process. It can be well explained that the agent considered $s_1$ and $s_{18}$ as important features because the reward function is defined using the states (Chapter 5.1). Also for s1, the rule is derived using internal node: $s_1 \leq 2.2℃$ (Figure 5-4) which can be interpreted as the agent learned the dynamics of the system and considered not only current rewards but also the future rewards. However, it is hard to interpret the reason that the agent chose actions based on $s_{12}$ and $s_{19}$. In other words, making decision trees using state-action pairs can be superficial interpretation for artificial intelligences. Although it is hard to fully understand the decision making process of the DQN agent, the result of this study is worthy enough in that it presented a new perspective of making control strategies for better building control.

# 6. Conclusion

## 6.1 Summary

Building controls are becoming complicated due to novel systems such as intermittent renewables, energy storage systems, and more. Reinforcement learning based control has been attracting attention as a technique to implement such complex building system control and demonstrated its potential to enhance building performance while addressing some limitations of other control techniques. Especially, RL can be a model-free manner control technique which can liberate researchers from tedious works of developing simulation models. Although RL-based control can be a model-free approach, unstable control actions during early training period and lack of interpretability are challenges that should be overcome to implement RL for real building. RL controller may cause unexpected costs (eg. thermal discomfort of occupants or failure of systems) by unstable control during early period of training. Also, buildings are owned and operated by humans who should be reconfirmed the controllers' intentions, especially regarding to failure cases. However, reinforcement learning based controllers are in black-box form, which makes the decision making process difficult to interpret.

To overcome above mentioned challenges, this study proposed federated model and interpreting decision making process of artificial agent using explainable reinforcement learning. Federated model is a novel concept of simulation model which consists of several modules developed using data from BEMS. As federated model is developed without any pre-defined topology rules, it could address limitations of conventional building simulation programs such as EnergyPlus and a simulation model could be developed with less subjective assumptions. Using the federated model, deep Q-network (DQN) is implemented to learn dynamics of a target building. The trained agent of DQN could save 31.9% of energy consumption compared to the baseline control while enhancing system stability by reducing system failure hours to 6 hours from 18 hours out of 42 hours. The result demonstrated that federated model could provide initial experiences for the DQN agent to develop its own control strategy which can enhance stability of RL based controller and avoid

unexpected outcomes during the early training period.

Decision tree which is an interpretable machine learning algorithm is applied to distill explanation of decision making process of the DQN agent. By training the decision tree using state-action pairs generated by the trained DQN agent, it is possible to interpret reasons of choosing actions for the agent. For example, the most important state for the agent to choose an action is temperature of ice-based thermal storage (ITS) outlet and the reference value is 2.23℃. As the agent is required to find the strategy that the temperature would not exceed 3℃ while operating ITS system, the DQN agent explored strategies that could keep the temperature below 3℃ continuously. In other words, the result showed that DQN agent could find strategies that meet the requirement by considering not only current rewards but also future rewards. Also, the explanation could be applied for building control as simple but practical 'If-then' rules. By applying the rules derived using decision tree, 29.1% of energy consumption could be saved compared to baseline control and system failure hours could be reduced to 7 hours from 18 hours. This indicated that well evaluated rules could perform as effective as complicated artificial intelligence controllers.

To sum up, this study proposed practical framework of implementing reinforcement learning based control for complicated building system. Also, it is demonstrated that explainable reinforcement learning can be applied to develop reduced rule whose performance is near RL controller and reliability is enhanced (Figure 6-1).

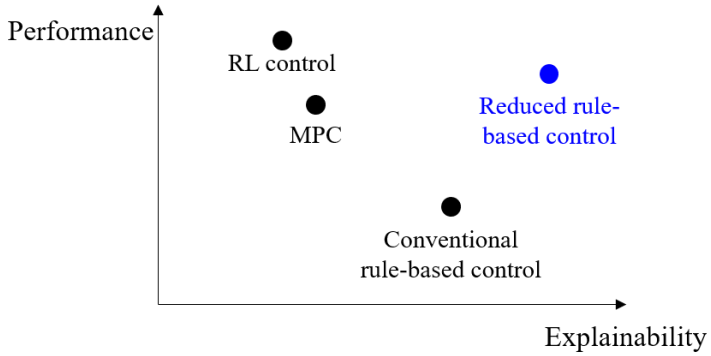



**Figure 6-1. Hypothesis of the study**

## 6.2 Future works

This study proposes rule reduction framework using explainable reinforcement learning and demonstrates that reduced rules can perform as well as complex reinforcement learning algorithms. The significance of this study lies in proposing how to derive rules with quantitative evaluation for building control. Followings are future studies:

- One of the major advantages of applying reinforcement learning controller is that RL-based controller can be adaptive. As the agent of RL learns dynamics of building by continuously interacting with the environment, the controller can adapt to internal and external changes. However, the reduced rule-based control cannot adapt by itself to the changing environment. Therefore, studies on developing adaptive rule-based controller should be conducted to conserve the advantage of RL-based controller while extracting simple rules.

- This study is conducted using data gathered in summer. Therefore, the developed controller could be implemented only in summer season. Also, the controller could perform well only for the target building because it is developed only using data gathered from the building. The studies on developing spatially and temporally generalizable controller should be conducted to implement the developed framework widely.

# References


Ahn, K. U. (2018), *Model-free Optimal Control for Building Energy System using Deep Q-learning*, Ph.D. Dissertation, Suwon, Sungkyunkwan Universtiy

Ahn, K. U., Kim, D. K., Kim, Y. J., Yoon, S. H., and Park, C. S., (2016), Issues to Be Solved for Energy Simulation of An Existing Office Building, *Sustainability*, 8, 345, doi:10.3390/su8040345

Alharin, A., Doan, T.N., and Sartipi, A.M. (2020), Reinforcement Learning Interpretation Methods: A Survey, *IEEE Access*, Vol. 8, pp. 171058-171077, doi: 10.1109/ACCESS.2020.3023394

Cacabelos, A., Eguia, P., Febrero, L., and Granada, E. (2017), Development of a new multi-stage building energy model calibration methodology and validation in a public library, *Energy and Buildings*, 146, pp.182-199

Chen, Y., Norford, L. K., Samuelson, H. W., and Malkawi, A. (2018), Optimal control of HVAC and window systems for natural ventilation through reinforcement learning, *Energy and Buildings*, 169, pp.195-205

Coppens, Y., Efthymiadis, K., Lenaerts, T., and Nowe, A. (2019), Distilling Deep Reinforcement Learning Policies in Soft Decision Trees, *Proc. of the IJCAI 2019 Workshop on Explainable Artificial Intelligence*, pp. 1-6

Ding, X., Du, W., and Cerpa, A. E. (2020), MB2C: Model-Based Deep Reinforcement Learning for Multi-zone Building Control, *Proc. of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pp.50-59

Ding, Z., Hernandez-Leal, P., Ding, G. W., Li, C., and Huang, R. (2021), CDT: Cascading Decision Trees for Explainable Reinforcement Learning, *arXiv:2011.07553v2*

DOE. (2015), Chapter 5: Increasing Efficiency of Building Systems and Technologies, *QUADRENNIAL TECHNOLOGY REVIEW: An Assessment of Energy Technologies and Research Opportunities*

DOE. (2020), EnergyPlus 9.3 Engineering Reference: The Encyclopedic Reference to EnergyPlus Calculations, *U.S. Department of Energy*

DOE. (2020), EnergyPlus 9.3 Input Output Reference: The Encyclopedic Reference to EnergyPlus Calculations, *U.S. Department of Energy*

Došilović, F. K., Brčić, M., and Hlupić, N. (2018), Explainable artificial intelligence: A survey, *Proc. of 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp.210-215, doi: 10.23919/MIPRO.2018.8400040

Dulac-Arnold, G., Mankowitz, D., and Hester, T. (2019), Challenges of Real-World Reinforcement Learning, *Proc. of the 36th International Conference on Machine Learning*, Long Beach, California, USA, PMLR 97

Ernst, D., Glavic, M., Capitanescu, F., and Wehenkel, L. (2009), Reinforcement Learning Versus Model Predictive Control: A Comparison on a Power System Problem, *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol.39 (2), pp.517-529

Fu, Y., Zuo, W., Wetter, M., VanGilder, J. W., and Yang, P. (2019), Equation-based object-oriented modeling and simulation of data center cooling systems, *Energy and Buildings*, 198, pp.503-319

Geyer, P. and Singaravel, S. (2018), Component-based machine learning for performance prediction in building design, *Applied Energy*, 228, pp.1439-1453

Hong, T., Langevin, J., and Sun, K. (2018), Building simulation: Ten challenges, *Building Simulation*, Vol. 11 (5), pp.871-898

Huang, Y., Seck, M. D., and Verbraeck, A. (2011), From data to simulation models: component-based model generation with a data-driven approach, Proc. of the 2011 Winter Simulation Conference, pp.3719-3729, doi:10.1109/WSC.2011.6148065

Kazmi, H., Mehmood, F., Lodeweyckx, S., and Driesen, J. (2018), Gigawatt-hour scale savings on a budget of zero: Deep reinforcement learning based optimal control of hot water systems, *Energy*, 144, pp.159-168

Kelmen, A., Ma, Y., and Borrelli, F. (2013), Analysis of local optima in predictive control for energy efficient buildings, *Journal of Building Performance Simulation*, 6:3, pp.236-255, doi:10.1080/19401493.2012.671959

Kim, Y. M., Ahn, K. U., and Park, C. S. (2016), Issues of Application of Machine Learning Models for Virtual and Real-Life Buildings, *Sustainability*, 8, 543, doi:10.3390/su8060543

Kim, Y. S. (2021), *Optimal Control of Lighting System based on Illuminance Prediction Model with Deep Deterministic Policy Gradient (DDPG)*, Master Dissertation, Seoul, Seoul National University

Lipton, Z. C. (2017), The Mythos of Interpretability, *arXiv:1606.03490v3*

Ma, Y., Borrelli, F., Hencey, B., Coffey, B., Bengea, S., and Haves, P. (2012), Model Predictive Control for the Operation of Building Cooling Systems, *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY*, Vol. 20 (3), pp.796-803

Madmual, P., Miller, T., Sonenberg, T., and Vetere, F. (2020), Distal Explanations for Model-free Explainable Reinforcement Learning, *arXiv:2001.10284v2*

Madmual, P., Miller, T., Sonenberg, T., and Vetere, F. (2020), Distal Explanations for Explainable Reinforcement Learning Agents, *arXiv:2001.10284v1*

Madmual, P., Miller, T., Sonenberg, T., and Vetere, F. (2019), Explainable Reinforcement Learning Through a Causal Lens, *arXiv:1905.10958v2*

Mnih, V., Kavukcuoglu, K., Silver, D. *et al.* (2015), Human-level control through deep reinforcement learning, *Nature* 518, pp. 529–533

Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A., and Brown, S. D. (2004), An introduction to decision tree modeling, *Journal of Chemometrics 2004* (18), pp. 275-285

Nagarathinam, S., Menon, V., Vasan, A., and Sivasubramaniam, A. (2020), MARCO – Multi-Agent Reinforcement learning based COntrol of building HVAC systems, *Proc. of 11th ACM International Conference of Future Energy Systems (e-Energy'20)*, Virtual Event, Australia, pp.57-67

Roth, A. M., Topin, N., Jamshidi, P., and Veloso, M. (2019), Conservative Q-Improvement: Reinforcement Learning for an Interpretable Decision-Tree Policy, *arXiv:1907.01180v1*

Sahlin, P. and Sowell, E. F. (1989), A Neutral Format for Building Simulation Models, *Proc. of IBPSA Building Simulation '89 conference*, Vancouver, Canada

Singaravel, S., Suykens, J., and Geyer, P. (2018), Deep-learning neural-network architectures and methods: Using component-based models in building-design energy prediction, *Advanced Engineering Informatics*, 38, pp.81-90

Wang, D., Yang, Q., Abdul, A., and Lim, B. Y. (2019), Designing Theory-Driven User-Centric Explainable AI, *Proc. of 2019 CHI Conference on Human Factors in Computing Systems*, ACM, Glasgow, Scotland, UK

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

Wetter, M. (2009), Modelica-based modelling and simulation to support research and development in building energy and control systems, *Journal of Building Performance Simulation*, Vol.2 (2), pp.143-161

Wetter, M., Bonvini, M., and Nouidui, S. (2016), Equation-based languages – A new paradigm for building energy modeling, simulation and optimization, *Energy and Buildings*, 117, pp.290-300

Yang, L., Nagy, Z., Goffin, P., and Schlueter, A. (2015), Reinforcement learning for optimal control of low exergy buildings, *Applied Energy*, 156, pp.577-586

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

Zhao, H., Zhao, J., Shu, T., and Pan, Z. (2021), Hybrid-Model-Based Deep Reinforcement Learning for Heating, Ventilation, and Air-Conditioning Control, *frontiers in Energy Research*, 8:610518, doi: 10.3389/fenrg.2020.610518

## 국문 초록

# 강화학습을 적용한 실용적인 건물 시스템 제어

조성권

건축학과 건축공학전공

서울대학교 대학원

HVAC 및 조명과 같은 기존 시스템과 간헐적 재생 에너지, 에너지 저장 시스템 등과 같은 새로운 시스템에도 대응해야 하므로 현대 건물 시스템 제어는 복잡해지고 있습니다. 이에 따라, 건물 시스템 제어기는 건물의 동적 거동에 스스로 적응할 수 있어야 하고 다목적 최적화 결과를 반영할 수 있어야 한다. 강화학습 (reinforcement learning, RL)을 사용하여 전술된 건물 제어기의 성능을 달성할 수 있다는 것은 널리 알려져 있지만, RL을 실제 건물에 적용하기 위해서는 해결해야 할 과제들이 있다: (1) RL의 초기 훈련 기간 동안 불안정한 제어는 예상치 못한 비용을 야기할 수 있다. (2) 여전히 대부분의 RL 기반 제어 전략은 일상적 실무에 적용하기에는 시설 관리자 입장에서 이해하기 어렵고 제어 전략에 대한 해석을 수행할 수 없다. RL 알고리즘을 건물 제어에 적용한다는 것은 의사결정의 주체가 인공지능이 된다는 것을 의미한다. 이때, 건물의 소유주와 운영자는 인공지능 기반 건물 제어기의 의도 및 의사결정 과정에 대한 해석 및 이해를 할 필요가 있다.

첫 번째 과제를 해결하기 위해, RL 에이전트를 사전 학습하고 이를 위해 새로운 개념의 시뮬레이션 모델인 연합 모델이 제안된다. 연합 모델은 빌딩 시스템을 물리적 인과 관계에 따라 모듈로 나누고 각 모듈을 데이터 기반 모델로 개발하여 빌딩 시스템에 대한 시뮬레이션을 수행하는 통합

데이터 기반 모델이다. 대상 건물의 냉방 시스템 시뮬레이션 모델은 6개의 모듈로 구성되고 각 모듈은 BEMS에서 수집된 데이터를 사용하여 개발된다. 연합 모델은 제1법칙 기반 시뮬레이션 모델의 한계 (예: 위상 규칙, 모델 보정)를 극복할 수 있다. Deep Q-Network (DQN)은 냉방 시스템의 동적 거동을 학습하고 건물에 냉방을 공급하는 동시에 에너지 사용을 줄일 수 있는 제어 전략을 모색하는 데 적용된다. DQN의 제어 성능을 현재 건물 운영자들이 적용하는 기존 제어 성능과 비교함으로써 RL 제어기가 시스템의 제어 효율성을 크게 개선할 수 있으며 연합 모델은 강화학습 기반 제어기의 학습을 위한 가상 환경을 제공할 수 있음을 증명한다.

DQN 에이전트의 해석성을 높이기 위해 의사결정 트리를 사용하여 에이전트의 의사결정 프로세스에 대한 설명을 추출한다. 에이전트에서 생성된 상태-작업 (state-action) 쌍이 의사결정 트리를 훈련하는 데 사용된다. 얕지만 쉽게 해석할 수 있는 모델을 사용한 사후 해석은 강화학습의 투명성과 해석성을 향상시킨다. 또한 의사결정나무가 만든 분류 결과는 인공지능이 만든 제어 전략을 단순화시킨 'If-then' 규칙을 도출한다. 추출된 규칙 (reduced rule) 기반 제어의 성능과 DQN 제어기의 성능을 비교하여 두 제어기 사이의 에너지 절약량 차이가 2.8%로 미미함을 보인다. 즉, 규칙 기반 제어가 충분한 성능을 보인다는 것을 증명한다.

본 연구는 기축 사무실 건물의 냉방 제어를 위한 설명 가능한 RL의 적용 방안에 대해 수행된다. 의사 결정 트리를 훈련된 DQN 에이전트에 적용한 다음 일련의 단순화된 제어 규칙을 도출한다. 이 연구는 설명 가능한 강화학습을 이용한 정량화된 규칙 도출 프레임워크를 제안하고, 복잡한 강화학습 알고리즘과 비교하여 단순하지만 정량적인 평가가 수행된 규칙이 충분한 성능을 보여줄 수 있음을 보여준다. 이 연구의

의의는 건물 통제에 대한 정량적 평가를 통해 규칙을 도출하는 방법을 제안하는 데 있다.

**주요어**: 설명 가능한 강화학습, 정량화된 규칙 도출, 정량적 규칙 기반 제어, DQN

**학번**: 2019-21675