A Reinforcement Learning Agent for Controlling Contamination Level in the Buildings

Rana Genedy and Mostafa Meimand
Advanced Machine Learning project Report
Virginia Tech
Blacksburg, VA 24060

May 5, 2022

Abstract

The indoor air quality inside the buildings has a significant effect on public health. The The American Society of Heating, Refrigerating and Air-Conditioning requires spaces to implement ventilation systems to dilute the contaminants (e.g., CO_2, and $PM_(2.5)$ produced by the occupants. However, the ventilation systems account for a substantial amount of the energy consumption. This study proposes a new and innovative tool that controls the ventilation systems in buildings to constrain the CO_2 level. This controller uses deep Q-learning approach to enhance the indoor air quality which consuming less energy. This proposed method was compared to two benchmark controllers, a fixed interval controller and a proportional–integral–derivative controller. The deep Q-learning controller outperformed the benchmark ones; it consumed less energy while maintaining a considerably lower CO_2 contamination level.

1 Problem statement

The indoor air quality (IAQ) inside the buildings has a significant effect on public health. In 2004, ASHRAE published a standard that requires highly occupied spaces to implement heating, ventilation, and air conditioning (HVAC) to dilute the contaminants produced by the occupants. According to the US Energy Information Administration, the building sector, HVAC systems, particularly, accounts for more than 33% of the energy consumption. This project aims to develop a reinforcement learning (RL) agent to enhance the IAQ while considering energy consumption in buildings. RL is based on agents which interact with the environment. The environment is the system that influences the agents' behavior. The agents learn from the reward and start attaining the actions that improve the reward; in our case, actions are turning the ventilation system Off/ON and states are the CO_2 contamination level in the building.

2 Analysis and Results

2.1 The room model

Very limited data available on the IAQ, hence, we used simulation approach to model the contamination and the CO_2 diffusion in buildings. CONTAM software was used for that simulation. The main idea behind CONTAM software is that it models a room with a source of contamination, a sensor, and an Air Handling System (AHS). To capture the dynamic behavior of the building, we randomly changed the HVAC system state (ON/OFF) for a month and recorded the CO_2 levels with each given action in a synthetic dataset.then, we developed a predictive model based on the generated dataset. Figure 2.1 shows the airflow versus CO_2 concentration. As expected the airflow and the contamination level are inversely proportional.

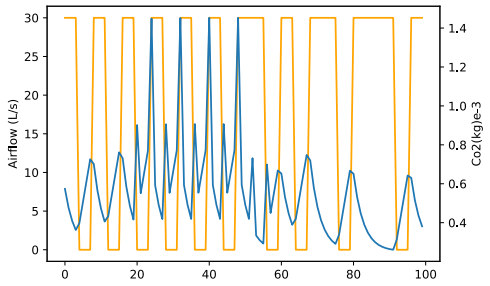

Figure 1: Airflow vs concentration

2.2 Two benchmark controllers

To comprehensively assess the performance of the proposed deep Q-learning (DQN) controller, we created two benchmark controllers that are commonly used in real life. The first one is the fixed interval controller and the second one is the proportional–integral–derivative controller (PID) controller.

- Fixed interval controller: In this controller implementation, we assume that there is no sensor in the room and the ventilation system is activated every fixed time interval based on the recommended air ventilation rate by the ASHRAE. Figure 2 show the performance of this controller in 60', 90' and 120' intervals. The CO_2 level should be between 400 - 1000. It appears that the only controller that can keep CO_2 within this range is the 60' controller.
- PID controller: In this controller, we assumed that there is a sensor that measures the CO_2 concentration in a room. Figure 3 show the performance of the proposed controller. This controller simply turns on the ventilation when the CO2 concentration is exceeding the upper threshold.

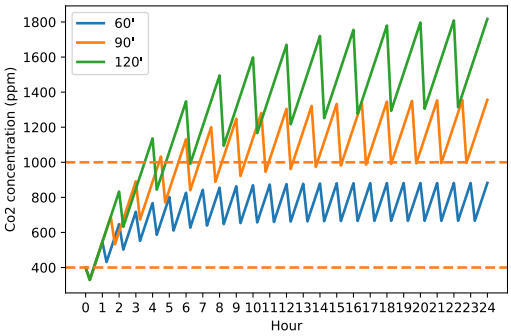

Figure 2: Without sensing controller

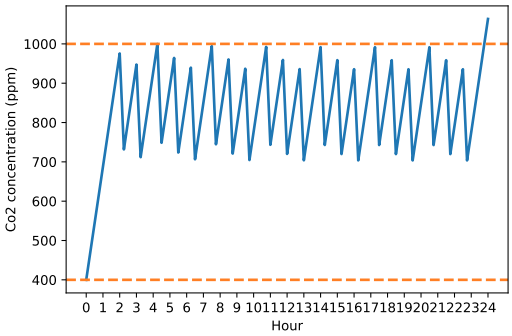

Figure 3: With sensing - PID

2.3 Co-Simulation

For validating the proposed controllers and since we cannot test them on a real test-bed, we employed CONTAM simulation. Figure 2.3 shows how this is implemented with CONTAM. briefly, at each timestep we calculate the estimated value of airflow and signal this value to CONTAM to get the new CO_2 level, then update the CO_2 level of the model.

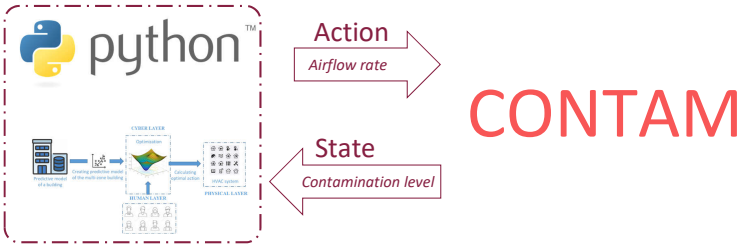

Figure 4: Co-Simulation framework

2.4 Deep neural networks

Figure 5 shows a schematic of the proposed method that incorporates the smart IAQ ventilation control system based on the DQN framework. DQN is a function approximation for the traditional Q-learning. traditional Q-learning uses Q-tables that require a huge amount of memory and usually take a large number of iterations to converge. In addition, traditional Q-learning trains in successive states; and learning directly from consecutive samples is inefficient, due to the strong correlations between these samples. Thus, there is a need for a function approximation to replace the Q-table and generalize across states. Therefore we proposed DQN. The main idea of DQN is that it addresses the problem of table memory, correlated data, and non-stationary distributions.The DQN was first introduced by Google's DeepMind [1]. The main attribute of the DQN is the experience replay mechanism known by the replay memory. During the learning process, the agent accumulates the transitions states which include the current state $s_(t)$, the action taken at , the new state resulted from the action $s_(t+1)$, and the corresponding reward of the selected action $r_(t+1)$. This transition state is stored in replay memory, and the replay memory is given a maximum size N. Hence, it does not store the whole history and it does not consume as much memory as the Q-table. Random mini-batches of transitions are sampled from the replay memory to train the model, therefore it reduces the samples correlation problem and results in a smoother training distribution over many past behaviors.

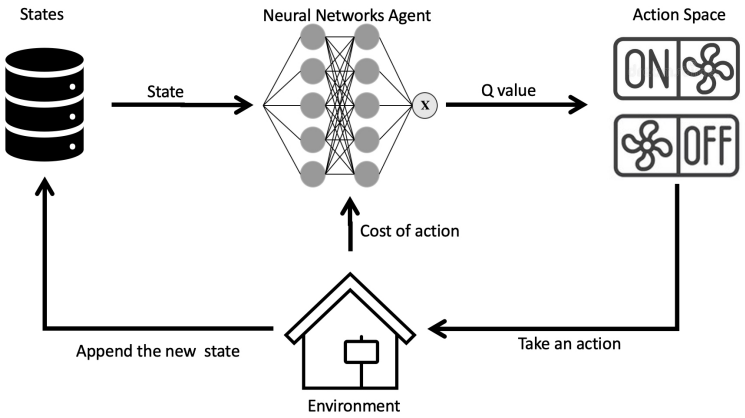

Figure 5: The implemented DQN framework

2.4.1 Environment implementation

The environment represents the room of interests and how to calculate the new value of the CO_2 at every time step. Equation 1 is used to calculate the value of CO_2 given the value of the CO_2 in the pervious step and the

ventilation rate. This equation was derived from the room model outputs calculated by CONTAM. The initial conditions of the environment are the initial CO_2 which is set to be the average outdoor CO_2 concentrations (350 ppm), and the initial ventilation rate is set to be zero (i.e., the ventilation system is off). Every state update in the environment is associated by a unique reward. We handcrafted our unique reward function to penalize the actions that make the CO_2 state exceeds the maximum allowable CO_2 level, which in our case is 1000 ppm. The reward function also penalize turning the ventilation ON since it consumes energy. Exceeding the CO_2 maximum level has higher negative weight than turning the ventilation ON, hence we are guiding the model to consume less energy (i.e., keep the ventilation off) as long as the CO_2 level is below 1000 ppm. In summary, the environment estimates the $s_(t+1)$ and $r_(t+1)$ given s_t and a_t and save these values as a transition state in the replay memory.

$$CO_2(new) = -110 * \text{ventilation power} + CO_2(current) + 50 \tag{1}$$

2.4.2 Neural network agent implementation

We used two identical Q approximators (i.e., neural networks) in this problem, a main and a target model. The main reason for having two neural networks is that the models start by taking a large number of random actions to explore the environment. Since we aim to make our Q predictions more consistent and stable. We train the model using the main neural network. Then, every multiple steps we update the target neural network weights to be the same as the main neural network and predict the new Q values using the target network. The two neural networks in this problem are fully connected feed forward neural networks. Each consists of four layers (i.e., one input layer and three hidden layers) composed of 128, 64, and 64 nodes for each hidden layer. The Python Keras library was used to build the neural networks.

2.4.3 Model training

The Neural network agent starts by populating the replay memory by taking random actions, applying them to the environment, and storing the transition states associated with that action. Once the replay memory size exceeds a certain threshold, which was set to 1000 steps, the training phase is initiated. In the training phase, the main neural network randomly samples a mini-batch of 32 samples from the replay memory to train, where training parameters are the states, and the corresponding labels are the Q-values. This goes on for a number of iterations, then the weights of the target network get updated to match the main neural network and predict the new Q-values. The prediction state occurs every 5 episodes, while every episode is one week. The selected action is controlled by the value of epsilon (ϵ) following ϵ-greedy approach. The initial epsilon was set to 0.9 and the minimum value of ϵ was assigned to 0.01 for continuous exploration of the environment.

2.4.4 The DQN output

Figures 6 and 7 show the DQN controller performance through the episodes. In early episodes when the value of ϵ is high, the neural network agent controls the environment unsteadily based on selecting random actions. Therefore, the agent was not able to learn from the causality between the ventilation rate and the CO_2 concentration. Progressively, the neural network agent learned how to select right actions to control the environment. For example, at the 4000 time step, the agent performance started to converge to be with the specified limit (1000 CO_2 ppm). Starting from time step 14000, the agent found the optimum policy to properly control the IAQ system. consecutively, The episodes rewards started very low and then as the policy improved, the reward kept improving and stabilizing. The reward converged to almost zero (the highest reward possible) on episode 1600.

3 Results discussion and conclusion

We compared the results of the proposed DQN controller with the two benchmark controllers. Figure 8 shows the proposed policy by the DQN agent over a day. The proposed action equals 1, then the ventilation is ON,

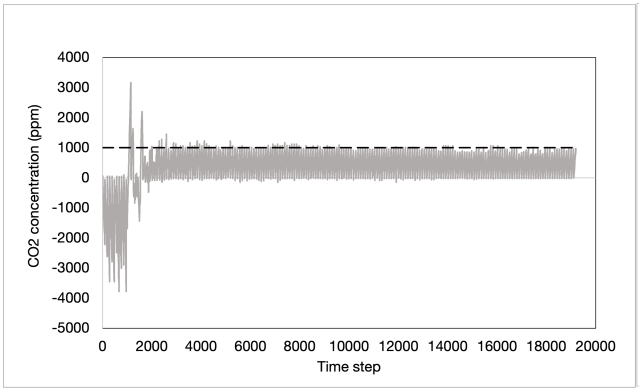

Figure 6: CO_2 concentration over time

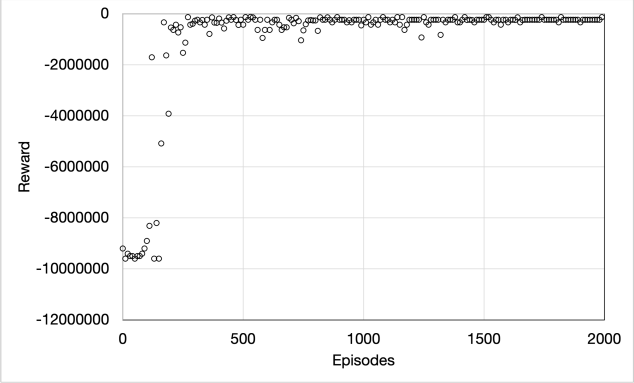

Figure 7: DQN reward overtime

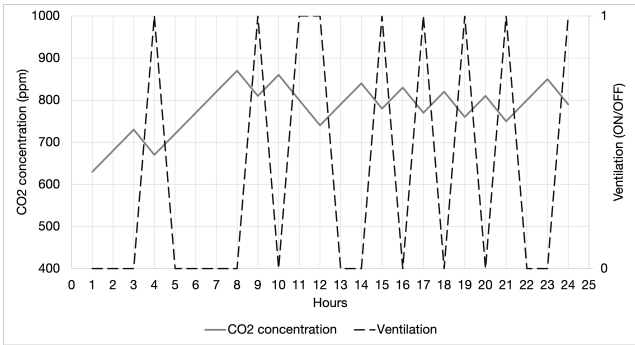

Figure 8: The DQN controller over a day

and equals zero when the ventilation is off. Although the virtual environment does not perfectly model the IAQ dynamics as CONTAM model, this figure shows that the DQN ventilation control system based more capable of managing the CO_2 level and reduce the airflow by an average of 55% compared to the benchmark controllers 2 and 3 . Finally the DQN controller keeps the average CO_2 level lower than the benchmark controller which promotes better health for the building occupants as shown in table 1.

Table 1: The output performance of the different proposed controllers

Method of sensing	**Airflow (L/s)**	**Average Co_2 concentration (ppm)**
Best without sensing	24	724
PID	20	831
DQN	9.75	691

4 Contributions

Both authors worked equally to develop the model and writing the manuscript.

References

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.