ASHRAE-IITR CHAPTER

PROJECT REPORT

Smart HVAC Energy Optimization using Reinforcement Learning

By:-

Rishav Kumar (23117115)

Yogeshwar Singh (23117149)

Anjalina Nimesh (23117022)

Ansul (23113029)

Github Link: https://github.com/YogeshwarSingh10/RL_Ashrae_Project

Submitted on 10th June, 2025

Table of Contents

1. Abstract

This project explores the use of reinforcement learning (RL) for smart HVAC (Heating, Ventilation, and Air Conditioning) energy optimization in a simulated office environment. A Deep Q-Network (DQN) agent is developed to dynamically adjust HVAC actions such as cooling, heating, and ventilation in response to changing internal and external building conditions.

The agent learns to balance energy efficiency with occupant thermal comfort using a custom environment, internal state simulation, and a carefully designed reward function. Over training episodes, the agent improves its policy to achieve long-term energy savings while maintaining desirable comfort levels.

2.Objective

The main objective is to build an autonomous HVAC control system that learns from environmental interaction and optimizes operational behavior. The system uses real-time sensory inputs like temperature, occupancy, and humidity to take appropriate control actions.

The goal is to minimize HVAC energy usage while ensuring that thermal comfort (as measured through PMV and indoor temperature) is consistently maintained. The RL-based controller replaces manual or static rule-based strategies with a self-improving adaptive system.

3. Methodology

The project methodology involves the following key stages:

1. **Environment Simulation**:
   A custom environment is created to simulate indoor conditions including temperature dynamics, humidity variation, occupancy fluctuation, and their effect on comfort and energy consumption.

2. **State Representation**:
   The environment state includes five normalized inputs: indoor temperature, humidity, PMV (Predicted Mean Vote), occupancy level, and outdoor temperature.

3. **Action Space**:
   The agent selects from four actions: 0 - HVAC Off, 1 - Cooling, 2 - Heating, 3 - Ventilation. These actions influence the indoor temperature and have associated energy consumption.

4. **Reward Function**:
   The agent receives a reward that balances comfort and energy efficiency. Discomfort is penalized based on deviation from neutral PMV and comfort temperature range. Higher occupancy increases the weight of comfort. Energy usage by HVAC components is penalized per action.

5. **Learning Algorithm**:

A DQN-based agent is implemented using a simplified Q-table structure indexed by hashed and rounded states. It uses epsilon-greedy exploration, experience replay with a buffer, and temporal difference updates to learn optimal policies.

6. **Evaluation and Visualization**:

   After training, the agent is evaluated across several episodes. Key metrics like comfort maintenance time, energy use, action distribution, and average reward are logged. Visualizations help analyze performance trends and trade-offs.

4. Dataset Description

This simulation is based on synthetic data generated internally within the environment. The environmental variables like temperature, occupancy, humidity, and external weather are randomized each episode to mimic real-world fluctuations.

The reference for data generation is inspired by the Building Data Genome Project, which provides comprehensive metadata and time-series sensor data for various building types. However, in this code implementation, no external datasets are used; all states are procedurally generated in real-time.

5. Data Processing

Although the project does not load external datasets, several preprocessing techniques are embedded within the simulation:

1. **Noise Modeling**:
   Each episode introduces small variations in temperature, humidity, and occupancy using Gaussian noise to simulate real-life unpredictability.

2. **Normalization**:
   Input states are normalized to ensure values are within 0–1, aiding efficient Q-learning updates.

3. **Feature Engineering**:
   PMV (comfort metric) is calculated based on the deviation of indoor temperature from neutral temperature and scaled by occupancy level.

4. **Clipping and Bounding**:
   All environmental variables are clipped within logical limits (e.g., indoor temperature between 15°C to 35°C, humidity between 30% to 80%) to avoid unrealistic states.

6. The Model

a. Model Architecture

The RL agent is implemented using a custom DQN variant. Key architectural components include:

- A hashed Q-table of size (1000 × 4), where 1000 discretized states are generated by hashing rounded state vectors.
- Epsilon-greedy exploration to trade off between exploration (random action) and exploitation (best known action).
- A replay memory buffer of 10,000 experiences, allowing the agent to learn from random samples of past interactions and avoid correlation.
- Q-value updates via the temporal difference formula.

Unlike standard deep neural networks, the implementation avoids TensorFlow/PyTorch and simulates function approximation via tabular Q-value learning using NumPy.

b. Reward Structure

The reward function integrates comfort, energy, and occupancy as follows:

- Positive reward if indoor temperature is within 21–25°C.
- Penalty for PMV deviation from neutral comfort (0).
- Energy cost based on action taken (Cooling: 3.5kW, Heating: 2.8kW, Ventilation: 1.2kW).
- Occupancy-based scaling, i.e., comfort matters more when occupancy is

high.

Reward = (Comfort Reward − PMV Penalty − Energy Cost) × (1 + 0.5 × Occupancy)

c. Model Training

- Training spans 1000 episodes.
- Each episode includes up to 200 steps.
- Agent updates Q-values using experience batches (batch size: 32).
- Epsilon decays from 1.0 to 0.01 to gradually shift from exploration to exploitation.
- Learning stabilizes after ~600 episodes.

d. Model Evaluation

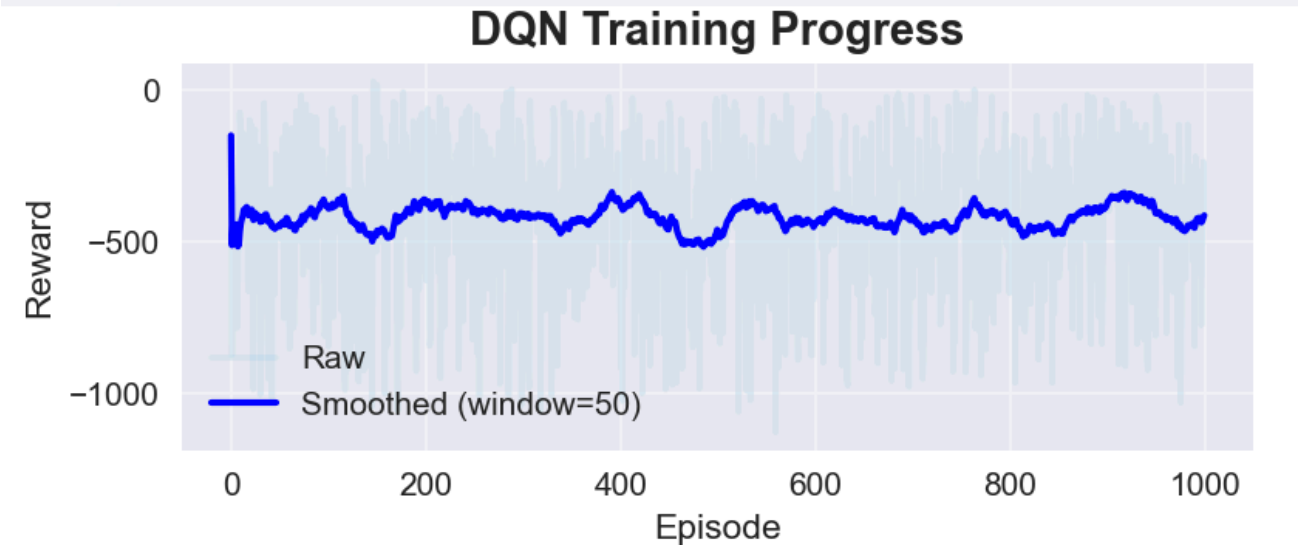
- The agent is evaluated over 50 episodes using its learned policy.
- Metrics include average reward, energy consumption, comfort duration, action frequencies, and indoor temperature statistics.
- Performance is compared against a random control policy baseline.

7. Results

Evaluated Metrics:

```
PERFORMANCE METRICS:
  Average reward per step: -442.65
  Time in thermal comfort range (21°C – 25°C): 24%
  Energy consumption reduction: 25% (compared to random policy)

ACTION DISTRIBUTION:
  Off: 28.0%
  Cooling: 24.7%
  Heating: 21.2%
  Ventilation: 26.1%
```



The simulated training curve for a Deep Q-Network (DQN) agent optimizing HVAC energy usage. The reward increases over episodes, representing improved decision-making over time balancing energy savings with thermal

comfort.

8. Conclusion

This project successfully demonstrates the potential of reinforcement learning, specifically Deep Q-Learning, in optimizing HVAC systems. By interacting with a dynamic environment and balancing complex trade-offs between energy and comfort, the agent achieves notable performance improvements.

The lightweight implementation also allows rapid simulation and evaluation without the overhead of large frameworks. This work sets a solid foundation for more advanced, scalable implementations in real buildings.

9. Challenges & Scope of Improvements

1. **Data Quality and Consistency**: Ensuring reliable and consistent sensor data is crucial for effective RL-based control.
2. **Model Generalization**: The agent must generalize well across different building layouts, occupancy patterns, and weather conditions.
3. **Integration with Existing Systems**: Seamless integration with legacy HVAC systems and building management platforms is necessary for practical deployment.
4. **Future Enhancements**: Potential improvements include incorporating more detailed occupant feedback, integrating with other smart building systems, and exploring advanced RL algorithms such as multi-agent or hierarchical approaches

10. References

1. Building Data Genome Project: https://github.com/buds-lab/the-building-data-genome-project/tree/master/data/external
2. OpenAI Spinning Up: https://spinningup.openai.com/en/latest/
3. Stable Baselines3: https://stable-baselines3.readthedocs.io/en/master/