RL-Based Control of Benchmark CSTR Processes with Safety Considerations

Seunghyun Cho, Hain Lee, Jaehyun Oh

May 18, 2025

1 progress

1.1 CSTR & Van de Vusse Reaction Code

The environment code and the training code have both been implemented, yet to be connected after modification. The environment code and the training code are mounted at GitHub Repository.

1.2 Code Details & Miscellaneous

For an initial testing, we ran a reference DQN training script instead of the training code we created. Fig 1 shows the performance of the DQN agent in the CSTR & Van de Vusse environment, illustrating that the environment is built well so that the agent is able to learn and improve its performance over time. However, since DQN does not support multi-action or continuous action spaces, we plan to implement a new version to support multi-action and continuous action spaces.

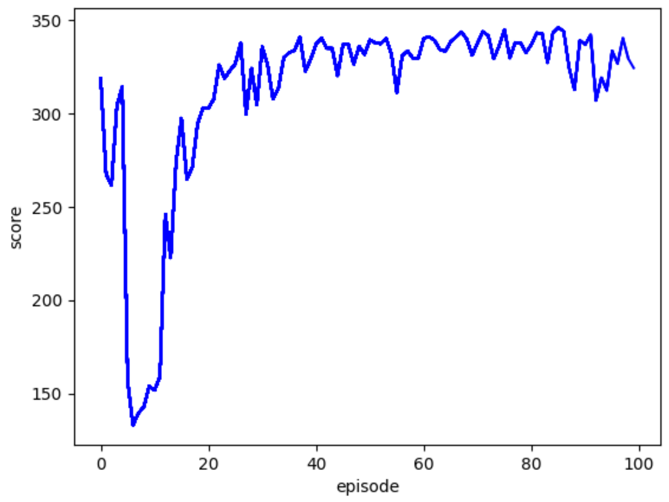

Figure 1: Defined states and actions in previous research

Reward shaping has not been performed yet. The current reward is represented in the form of a differential expression, such as $\frac{dC}{dt}$. The simulation is set in discrete time steps.

The action space is designed to include two variables: changes in feed flow and changes in heat, and feed flow change is represented as a list of five discrete values: +2, +1, 0, -1, and -2. However, the heat component is yet to be written in code, since it is hard to implement within the DQN training code. This will be addressed and verified within the training code.

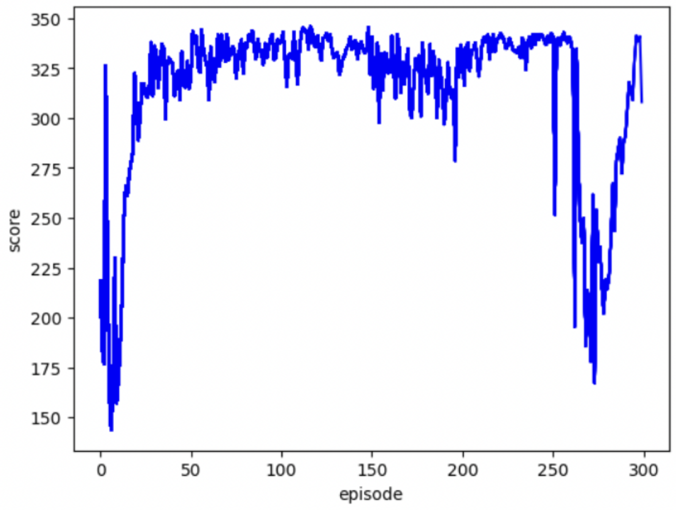

Figure 2: Defined states and actions in previous research

In Fig 2, we have observed that the sliding window (replay buffer) sometimes leads to a loss of important early informations, causing performance degradation in the later part of the training. Since the batch size is defined in terms of time steps and usually covers the entire episode, this is not a critical issue. However, if crucial data—such as failure events—are forgotten, it could negatively affect learning.

2 Preliminary

The current action space is discrete, which makes it compatible with DQN. However, DQN is inherently limited to discrete action spaces and cannot handle continuous actions. We plan to extend the implementation to support DQN, DDPG, and PPO. First of all, we are considering

DDPG, which supports continuous action spaces and allows for more complex, higher-dimensional actions. PPO will be implemented, as it can accept an action array directly. Attempting to compress multiple actions into a form compatible with DQN can lead to errors, so we aim to adopt PPO or DDPG to overcome this limitation and improve overall model flexibility.

We plan to implement reward shaping to enhance the learning process and ensure that the agent receives more informative feedback during training.

References