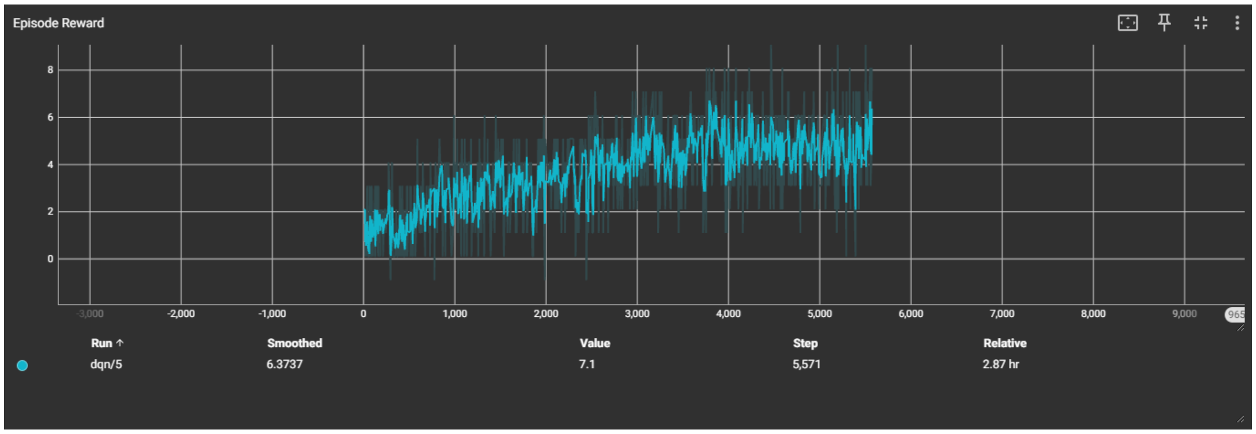
Solving Crafter – Assignment
Advanced artificial intelligence techniques
Adrian Dinu Urse

1. Value-Based Methods

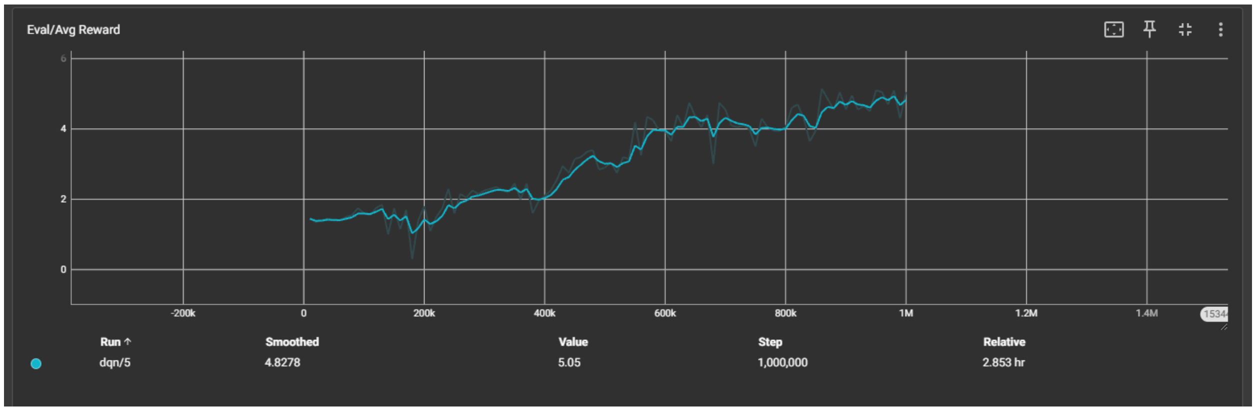
1.1 Deep Q learning Agent

Training paremeters:
train_steps = 1M
warmup_steps = 50.000
Replay Memory = 200.000
target_update_interval = 10.000
batch_size = 64
eval_interval = 10.000
lr = 1e-5
gamma = 0.95
epsilon_schedule = 800.000



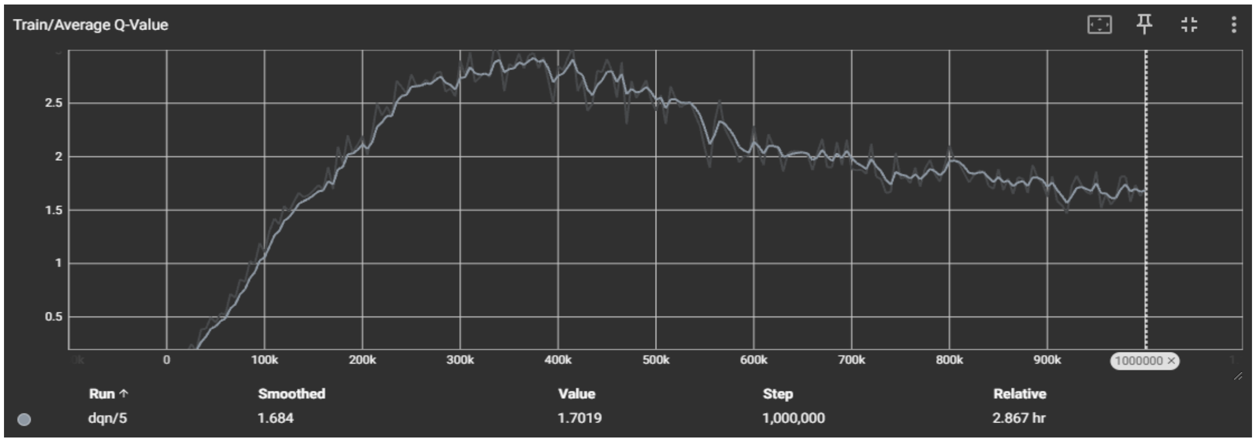
Run ↑	Smoothed	Value	Step	Relative
dqn/5	6.3737	7.1	5,571	2.87 hr

Episode reward over 1M steps



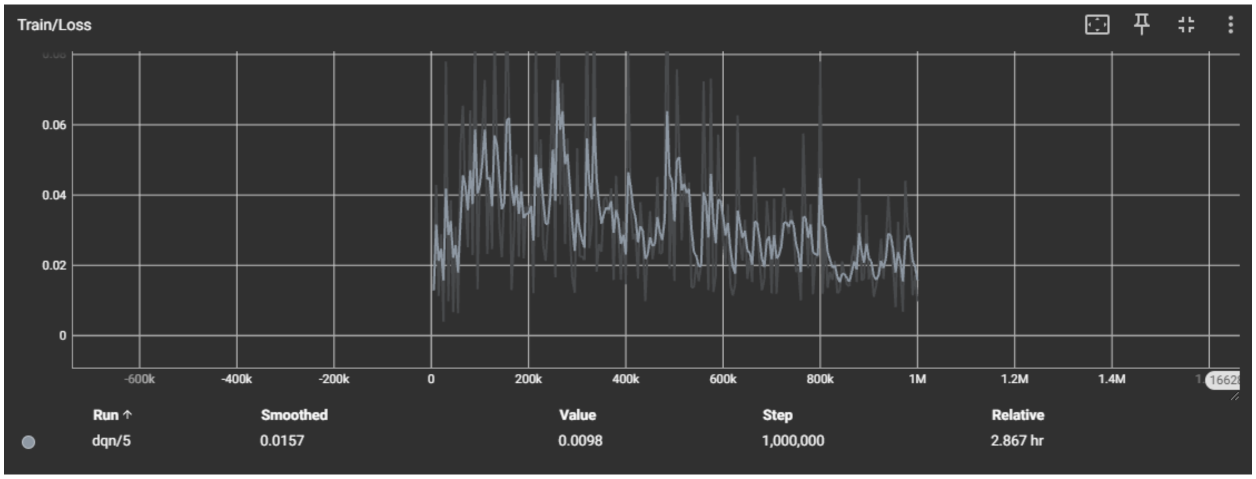
Run ↑	Smoothed	Value	Step	Relative
dqn/5	4.8278	5.05	1,000,000	2.853 hr

Evaluation average rewards



Run ↑ Smoothed Value Step Relative
dqn/5 1.684 1.7019 1,000,000 2.867 hr

Training Average Q-values over 1M steps



Run ↑ Smoothed Value Step Relative
dqn/5 0.0157 0.0098 1,000,000 2.867 hr

Training losses over 1M steps

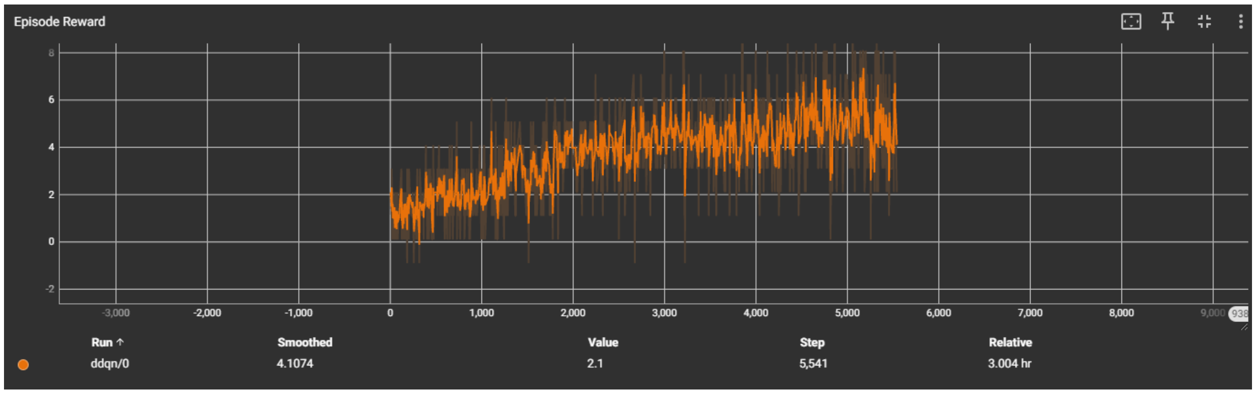
Final evaluation on 100 episodes:

Average Reward over 100 episodes: 5.62
Min Reward: 1.10
Max Reward: 10.10 Std Dev: 1.862

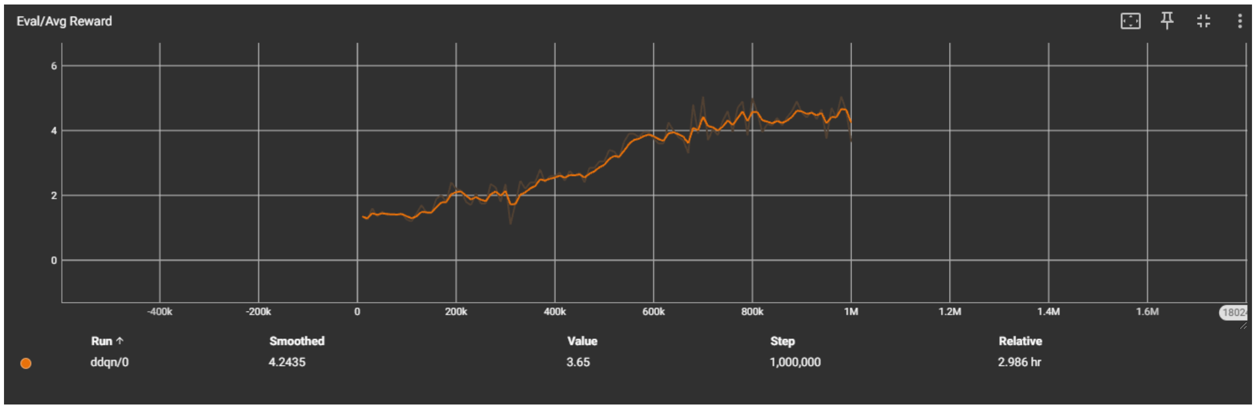
The implementation follows the DQN implementation from the provided lab resources. The objective function is to minimize the temporal difference (TD) loss. This aligns predicted Q-values with target values computed using the Bellman equation.

1.2 Double Deep Q learning Agent

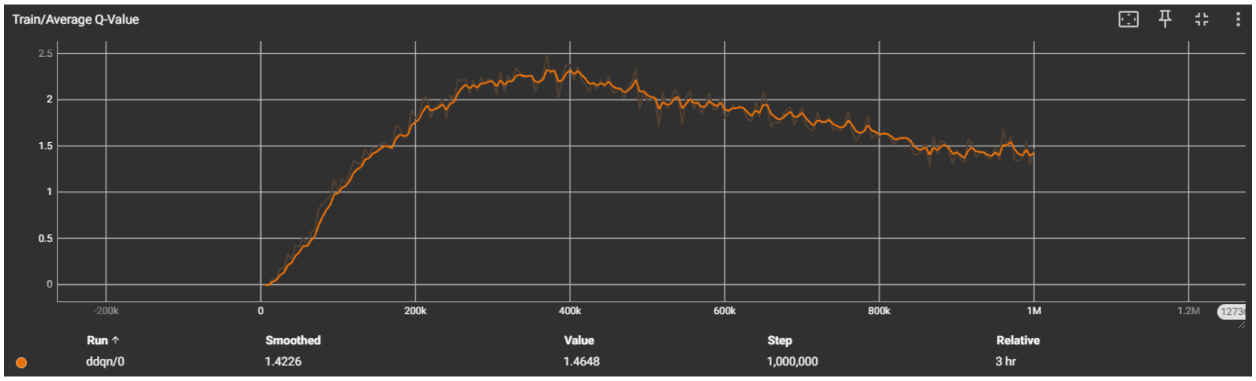
Training parameters:
train_steps = 1M
warmup_steps = 50.000
Replay Memory = 200.000
target_update_interval = 10.000
batch_size = 64
eval_interval = 10.000
lr = 1e-5
gamma = 0.95
epsilon_schedule = 800.000



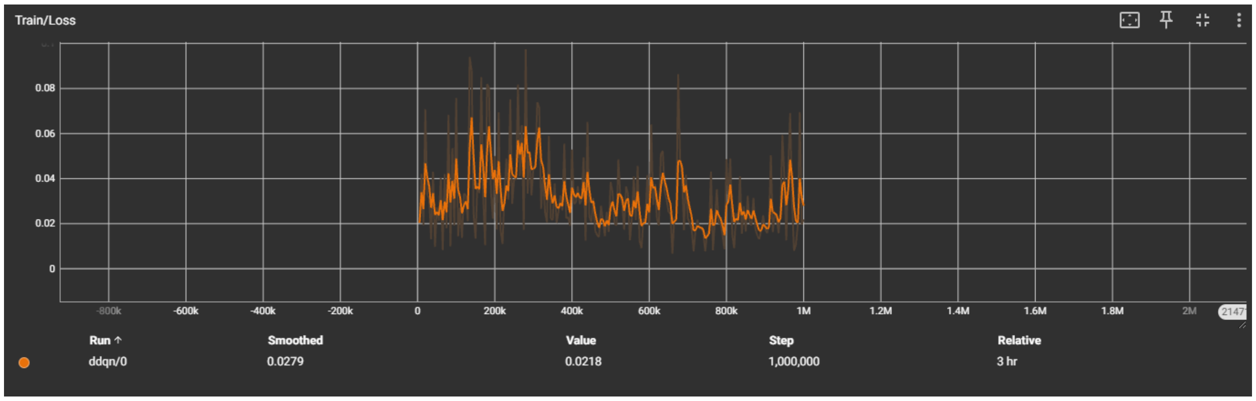
Episode reward over 1M steps



Evaluation average rewards



Training Average Q-values over 1M steps



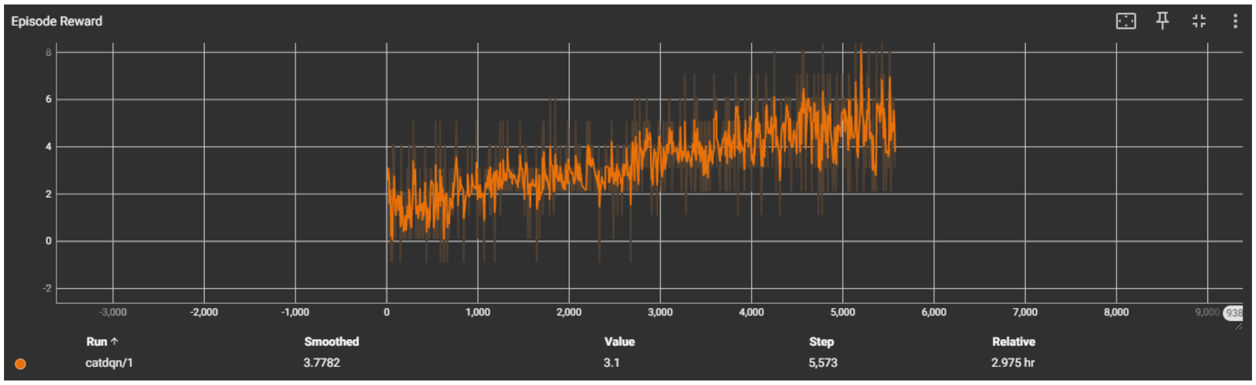
Training losses over 1M steps

Final evaluation on 100 episodes:
Average Reward over 100 episodes: 4.73
Min Reward: 1.10, Max Reward: 10.10, Std Dev: 1.847

I followed the DDQN implementation from the provided resources. Similar to DQN, its objective function minimizes the temporal difference (TD) loss to align predicted Q-values with target values derived from the Bellman equation. The key difference is that DDQN reduces overestimation bias by using the main Q-network for action selection and the target network for action evaluation, which improves stability.

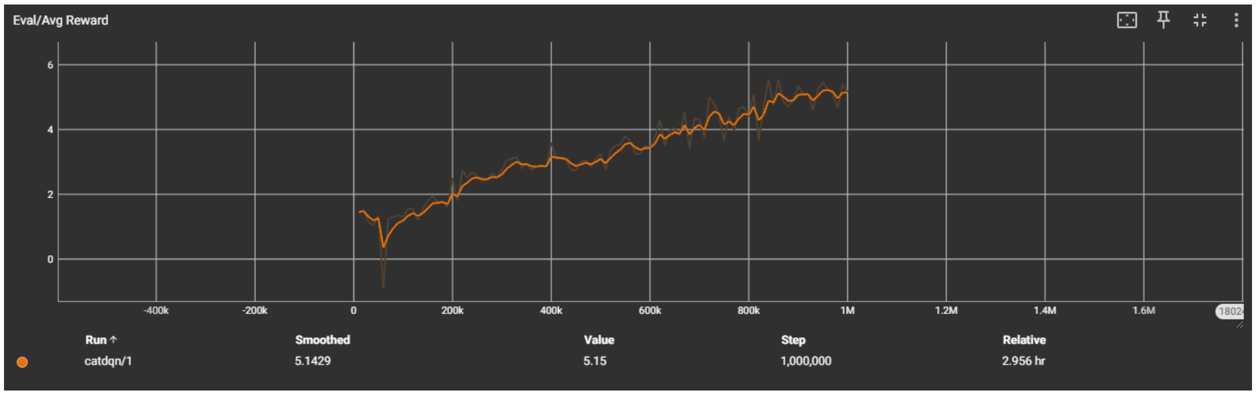
1.3 Categorical Deep Q learning Agent

Training paremeters:
train_steps = 1M
warmup_steps = 50.000
Replay Memory = 200.000
target_update_interval = 10.000
batch_size = 64
eval_interval = 10.000
lr = 1e-5
gamma = 0.95
epsilon_schedule = 800.000
V_MIN = -1
V_MAX = 23
NUM_ATOMS = 51



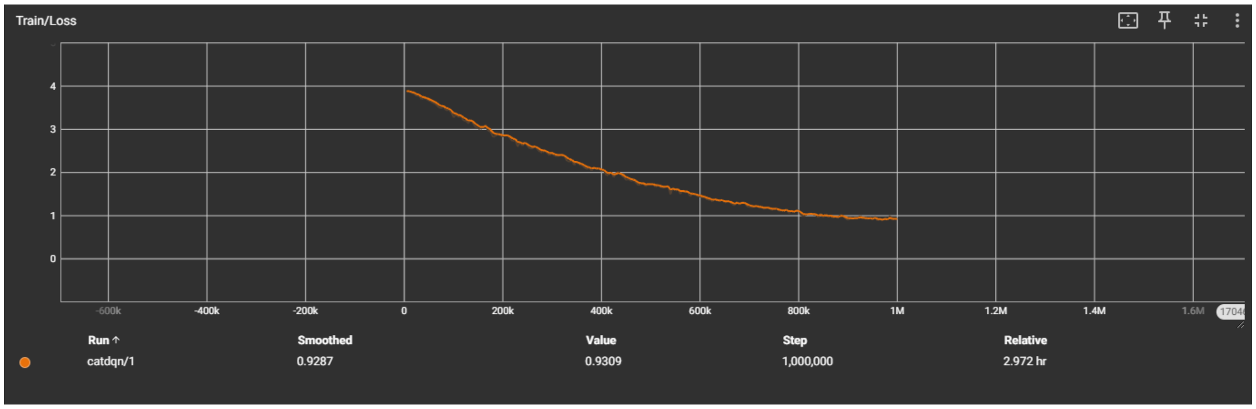
Run	Smoothed	Value	Step	Relative
● catdqn/1	3.7782	3.1	5,573	2.975 hr

Episode reward over 1M steps



Run	Smoothed	Value	Step	Relative
● catdqn/1	5.1429	5.15	1,000,000	2.956 hr

Evaluation average rewards



Run ↑	Smoothed	Value	Step	Relative
● catdqn/1	0.9287	0.9309	1,000,000	2.972 hr

Training losses over 1M steps

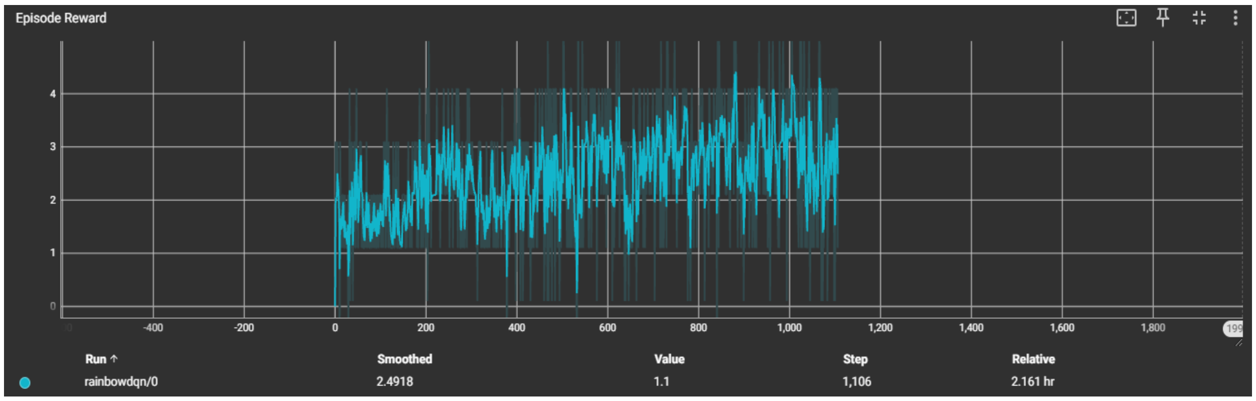
Final evaluation on 100 episodes:
Average Reward over 100 episodes: 4.79
Min Reward: 1.1, Max Reward: 9.10, Std Dev: 1.641

This implementation of Categorical DQN (C51) aims to predict a distribution of Q-values over discrete support atoms for each action, enabling a more granular representation of action values. During training, the agent minimizes the Kullback-Leibler (KL) divergence between the projected target distribution and the predicted distribution, effectively aligning the predicted distribution to expected returns.

The objective function in Categorical DQN is the minimization of the KL divergence, represented by the cross-entropy loss in the `update` function. This approach refines the typical DQN target by leveraging distributional information, capturing a wider range of potential outcomes rather than a single Q-value.

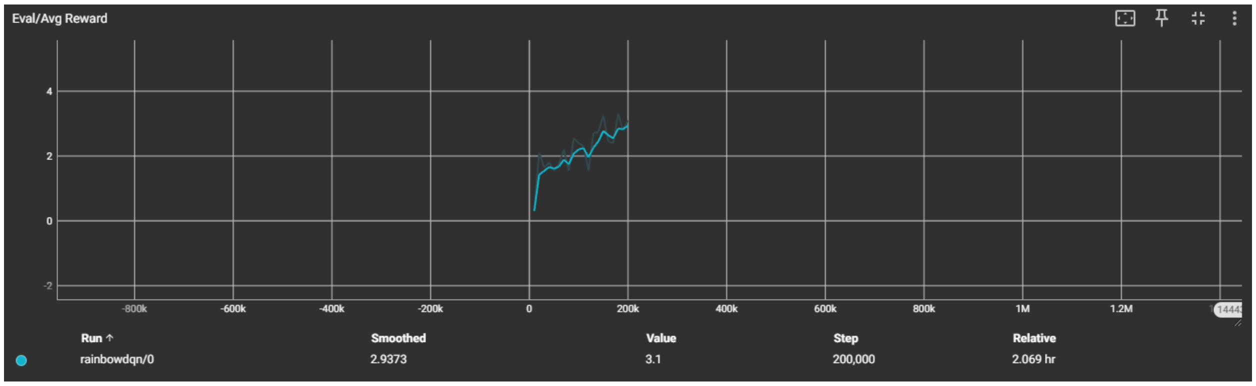
1.4 Rainbow Deep Q learning Agent

Training paremeters:
train_steps = 200k
Prioritized Replay Memory = 200.000
target_update_interval = 10.000
batch_size = 64
eval_interval = 10.000
lr = 1e-4
gamma = 0.99
n_steps = 3
V_MIN = -1
V_MAX = 23
NUM_ATOMS = 51
alpha=0.6
beta_start=0.4
beta_end=1.0
beta_anneal_steps=1.000.000



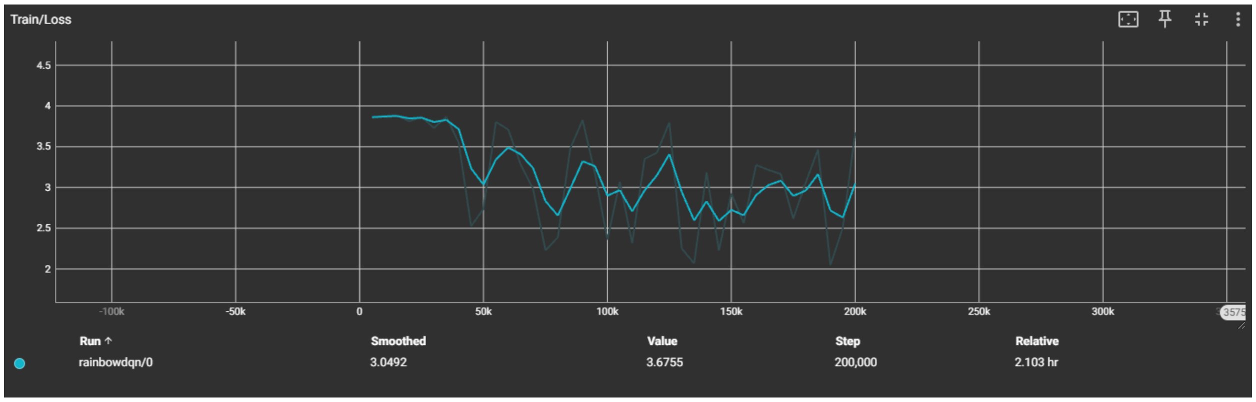
Run ↑	Smoothed	Value	Step	Relative
● rainbowdqn/0	2.4918	1.1	1,106	2.161 hr

Episode reward over 200k steps



Run ↑	Smoothed	Value	Step	Relative
● rainbowdqn/0	2.9373	3.1	200,000	2.069 hr

Evaluation average rewards



```
Run ↑          Smoothed   Value    Step      Relative
rainbowdqn/0   3.0492     3.6755   200,000   2.103 hr
```

Training losses over 200k steps

Final evaluation on 100 episodes:
Average Reward over 100 episodes: 2.55
Min Reward: 0.1, Max Reward: 5.1, Std Dev: 1.251

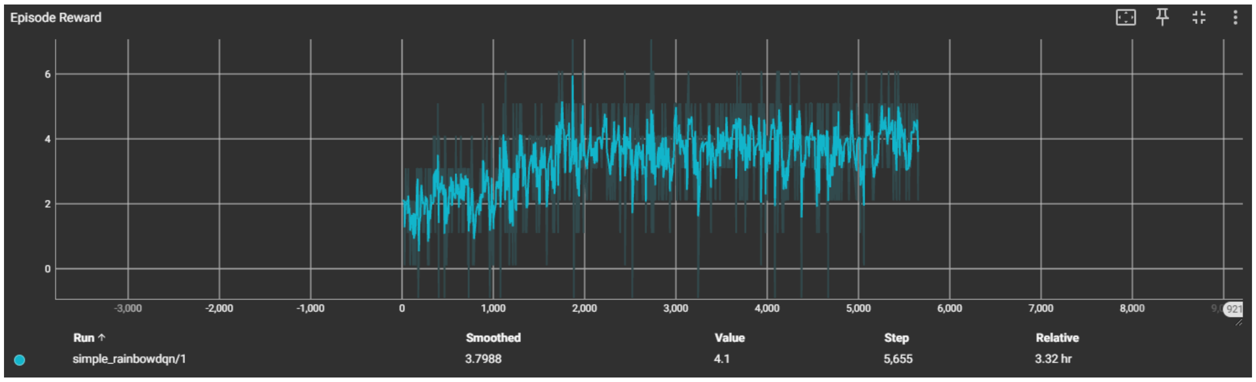
The implementation of Rainbow DQN combines several advanced techniques to enhance the stability and performance of the agent. The key components integrated here are:

1. Dueling Architecture: The network is split into value and advantage streams, allowing it to better differentiate between valuable and redundant actions in each state.
2. Noisy Layers: Noisy linear layers replace epsilon-greedy exploration with stochastic exploration by injecting noise directly into the network's weights, allowing the agent to explore more efficiently.
3. Distributional Q-learning: Instead of predicting a single Q-value, the network predicts a distribution of possible Q-values over discrete atoms, representing different possible outcomes.
4. Multi-Step Learning: Rewards are accumulated over multiple steps, allowing the agent to consider future outcomes more effectively and potentially accelerating learning.
5. Prioritized Experience Replay: Experiences are sampled with higher probability if they have higher TD errors, allowing the agent to focus on learning from more significant experiences.

The objective function of Rainbow DQN is to minimize the KL divergence between the projected target distribution and the predicted distribution. This is achieved by computing the categorical cross-entropy loss between the projected and the predicted distribution, weighted by importance sampling weights. This approach captures a wider range of possible rewards, providing a more robust learning signal.

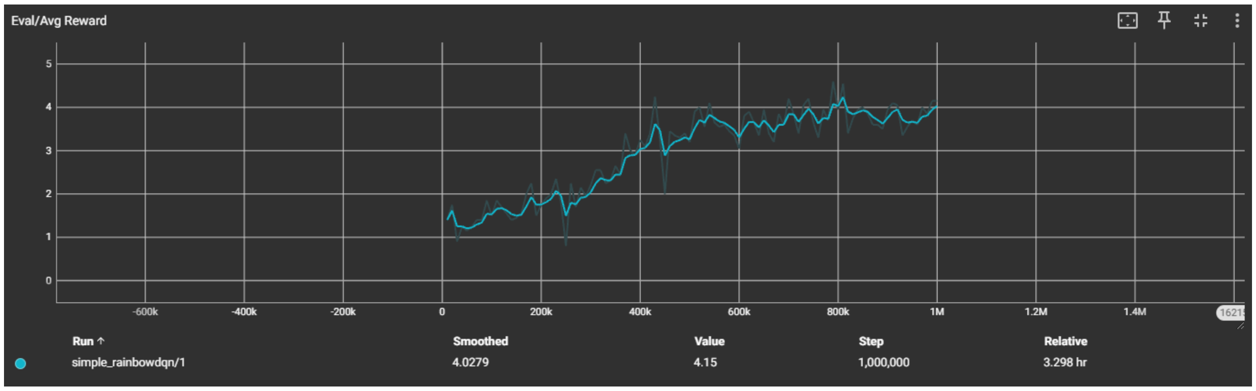
1.5 Simplified Rainbow Deep Q learning Agent

Training paremeters:
train_steps = 1M
Replay Memory = 200.000
target_update_interval = 10.000
batch_size = 64
eval_interval = 10.000
lr = 5e-5
gamma = 0.99
epsilon_schedule = 700.000
V_MIN = -1
V_MAX = 23
NUM_ATOMS = 51



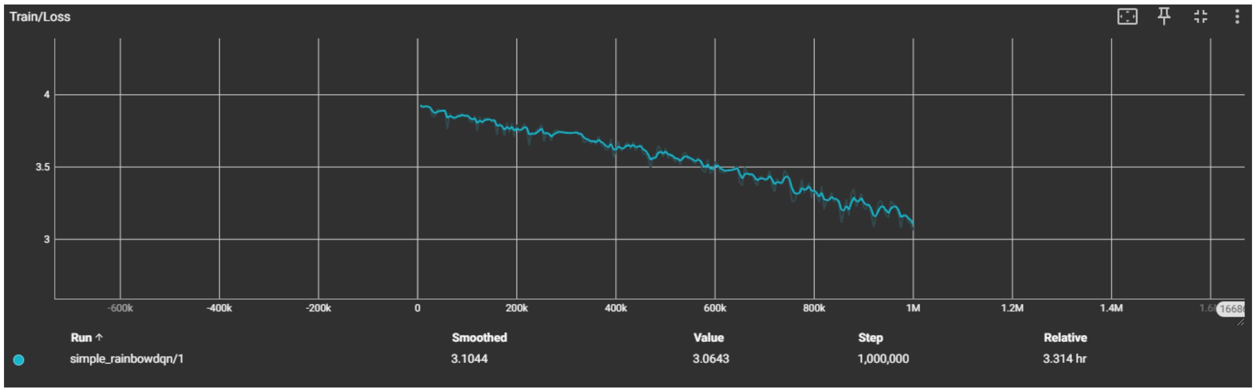
Run ↑	Smoothed	Value	Step	Relative
● simple_rainbowdqn/1	3.7988	4.1	5,655	3.32 hr

Episode reward over 1M steps



Run ↑	Smoothed	Value	Step	Relative
● simple_rainbowdqn/1	4.0279	4.15	1,000,000	3.298 hr

Evaluation average rewards



Run ↑	Smoothed	Value	Step	Relative
● simple_rainbowdqn/1	3.1044	3.0643	1,000,000	3.314 hr

Training losses over 1M steps

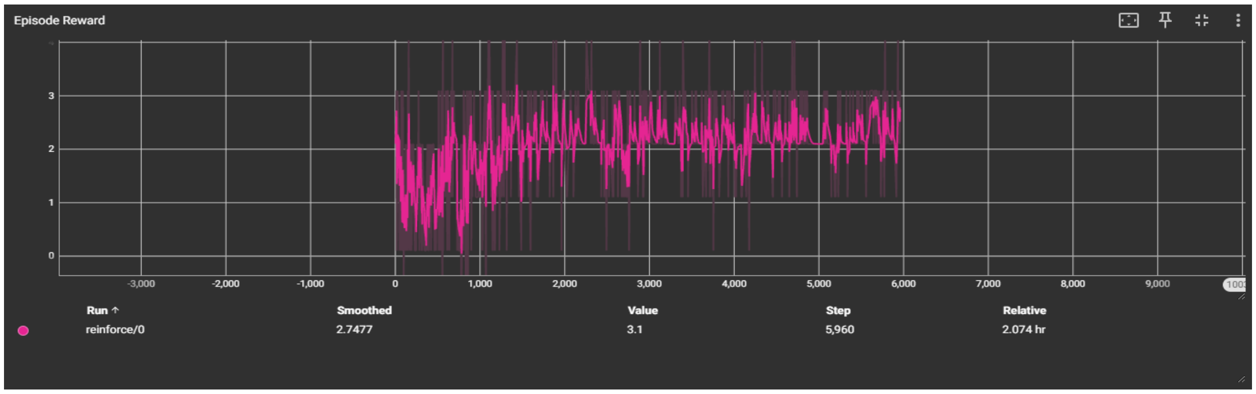
Final evaluation on 100 episodes:
Average Reward over 100 episodes: 4.19
Min Reward: 0.1, Max Reward: 6.1, Std Dev: 1.13

This simplified Rainbow DQN agent retains the core ideas of Rainbow DQN but removes noisy layers, prioritized replay, and n-step learning. It uses a dueling network architecture and a distributional approach to Q-value approximation, relying on Categorical DQN.

2. Policy-Based Methods

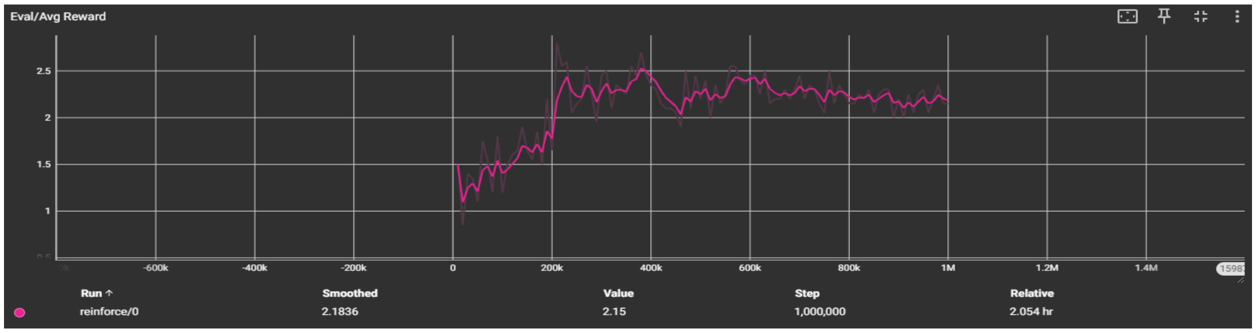
2.1 Reinforce Agent

Training paremeters:
train_steps = 1M
Replay Memory = 200.000
batch_size = 64
lr = 1e-5
gamma = 0.99



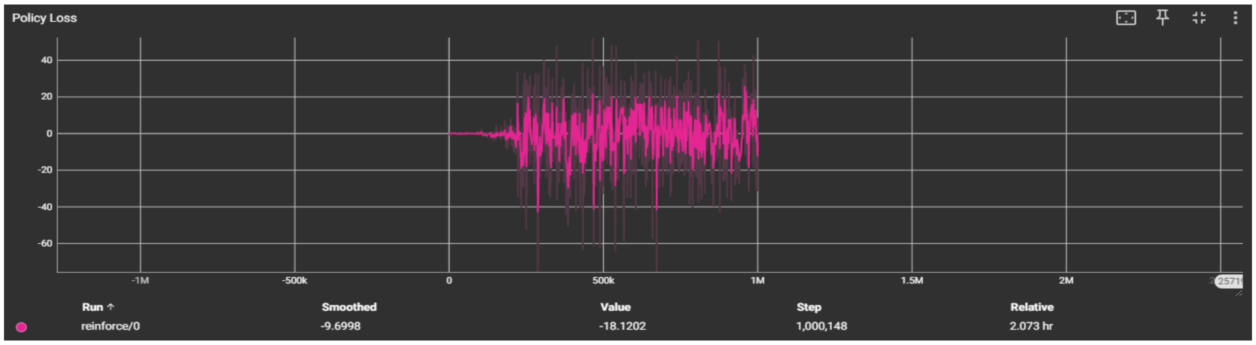
Run ↑	Smoothed	Value	Step	Relative
● reinforce/0	2.7477	3.1	5,960	2.074 hr

Episode reward over 1M steps



Run ↑	Smoothed	Value	Step	Relative
● reinforce/0	2.1836	2.15	1,000,000	2.054 hr

Evaluation average rewards



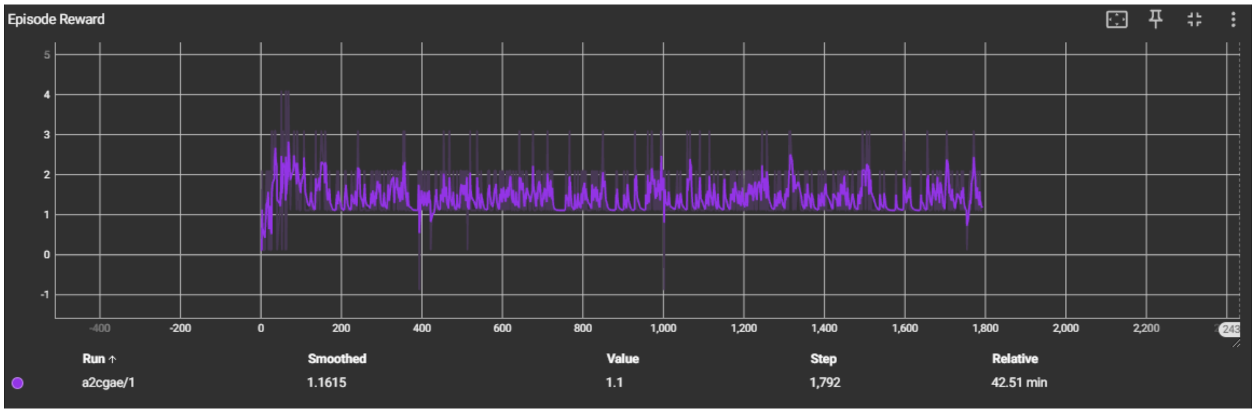
Run ↑	Smoothed	Value	Step	Relative
● reinforce/0	-9.6998	-18.1202	1,000,148	2.073 hr

Training losses over 1M steps

This Reinforce agent uses a CNN-based policy network to output action probabilities. The agent samples actions, logs probabilities, and records rewards at each step. After each episode, it calculates normalized discounted returns to stabilize learning. The objective is to maximize cumulative rewards by minimizing the policy loss, computed as the negative log-probabilities of actions weighted by returns. This encourages higher-reward actions through gradient descent.

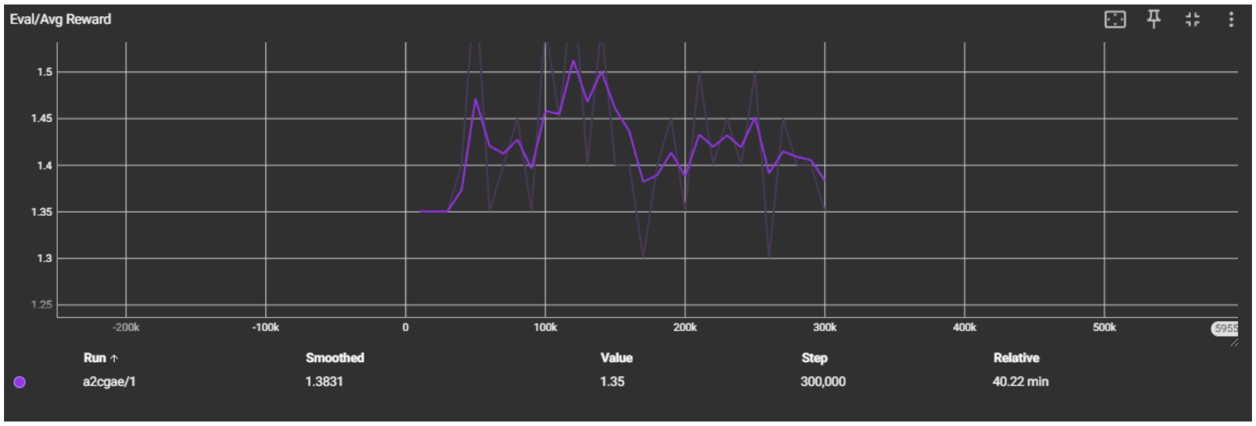
2.2 A2C-GAE Agent

Training paremeters:
train_steps = 300k
Replay Memory = 200.000
batch_size = 64
lr = 1e-5
gamma = 1e-5
beta = 0.01
tau = 0.95



Run ↑	Smoothed	Value	Step	Relative
a2cgae/1	1.1615	1.1	1,792	42.51 min

Episode reward over 300k steps



Run ↑	Smoothed	Value	Step	Relative
a2cgae/1	1.3831	1.35	300,000	40.22 min

Evaluation average rewards

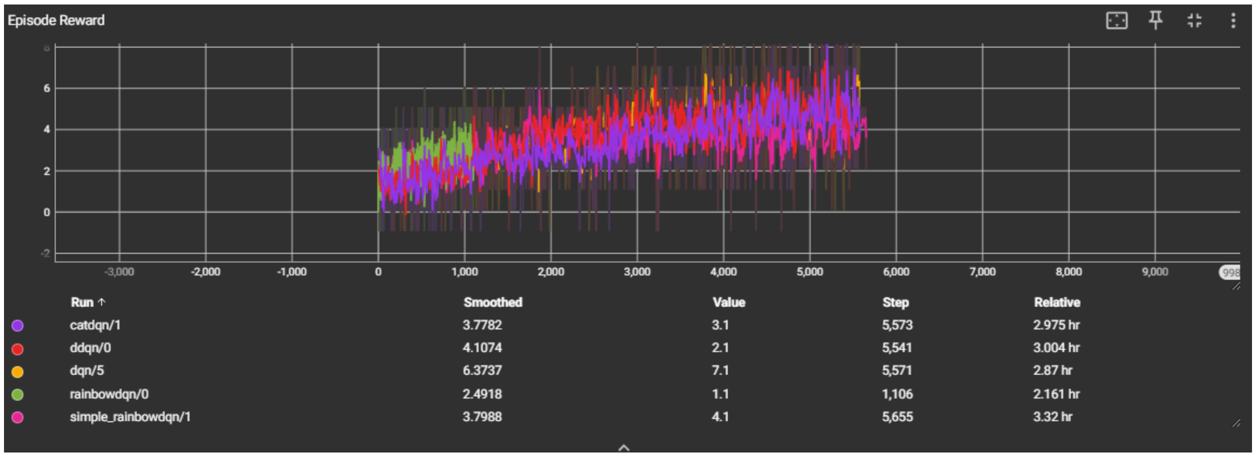
The A2C-GAE agent builds upon the Advantage Actor-Critic (A2C) approach by adding Generalized Advantage Estimation (GAE), which improves variance reduction when calculating returns. The agent uses a policy network to output action probabilities and value estimates for the states. The agent collects log probabilities, values, and entropy of each action to compute a return estimation.

Objective function: The agent optimizes a combined loss:

1. Policy loss: Based on the advantage (difference between returns and values), aiming to increase the probability of advantageous actions.
2. Critic loss: Reduces the error between estimated values and observed returns.
3. Entropy loss: Encourages exploration by maximizing entropy.

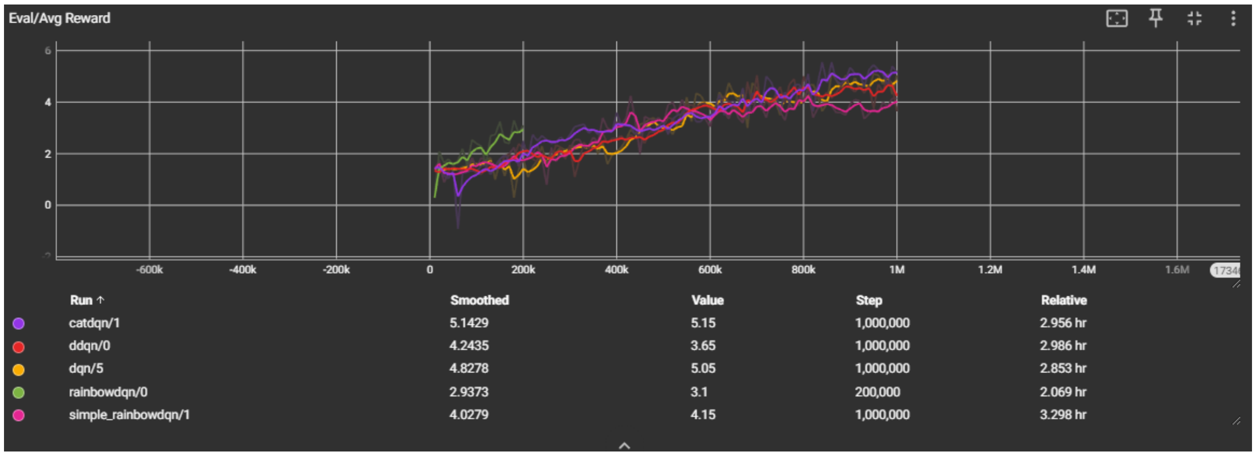
GAE refines the advantage estimation by blending multi-step returns, improving stability.

3. Value Based Methods Comparison



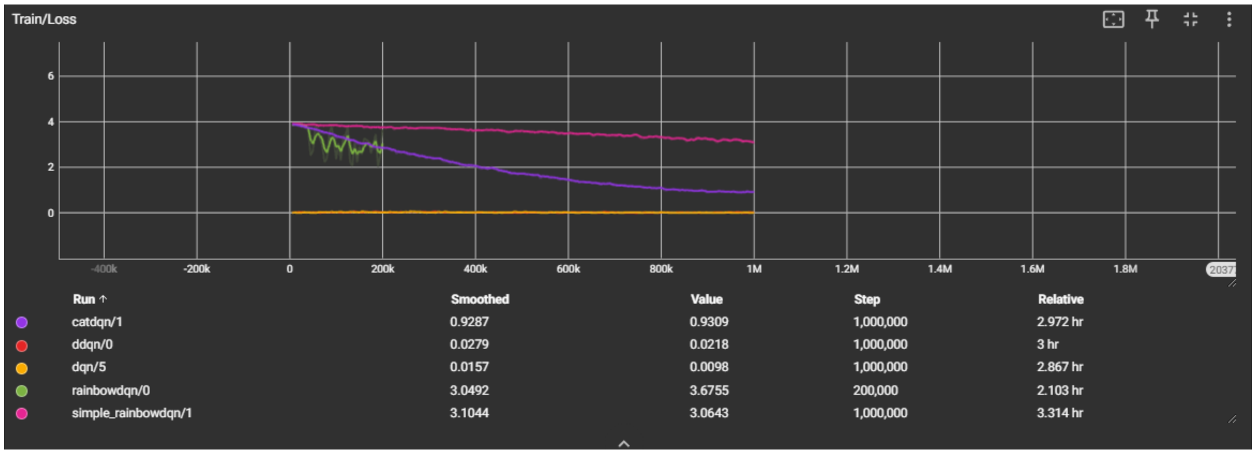
	Run ↑	Smoothed	Value	Step	Relative
●	catdqn/1	3.7782	3.1	5,573	2.975 hr
●	ddqn/0	4.1074	2.1	5,541	3.004 hr
●	dqn/5	6.3737	7.1	5,571	2.87 hr
●	rainbowdqn/0	2.4918	1.1	1,106	2.161 hr
●	simple_rainbowdqn/1	3.7988	4.1	5,655	3.32 hr

Episode rewards across all models



	Run ↑	Smoothed	Value	Step	Relative
●	catdqn/1	5.1429	5.15	1,000,000	2.956 hr
●	ddqn/0	4.2435	3.65	1,000,000	2.986 hr
●	dqn/5	4.8278	5.05	1,000,000	2.853 hr
●	rainbowdqn/0	2.9373	3.1	200,000	2.069 hr
●	simple_rainbowdqn/1	4.0279	4.15	1,000,000	3.298 hr

Evaluation average rewards across all models



	Run ↑	Smoothed	Value	Step	Relative
●	catdqn/1	0.9287	0.9309	1,000,000	2.972 hr
●	ddqn/0	0.0279	0.0218	1,000,000	3 hr
●	dqn/5	0.0157	0.0098	1,000,000	2.867 hr
●	rainbowdqn/0	3.0492	3.6755	200,000	2.103 hr
●	simple_rainbowdqn/1	3.1044	3.0643	1,000,000	3.314 hr

Training losses across all models

All agents show a generally upward trend in average reward as training progresses, indicating that they are all learning and improving over time.

Categorical DQN achieves the highest smoothed reward (5.15) and maintains a relatively consistent performance.

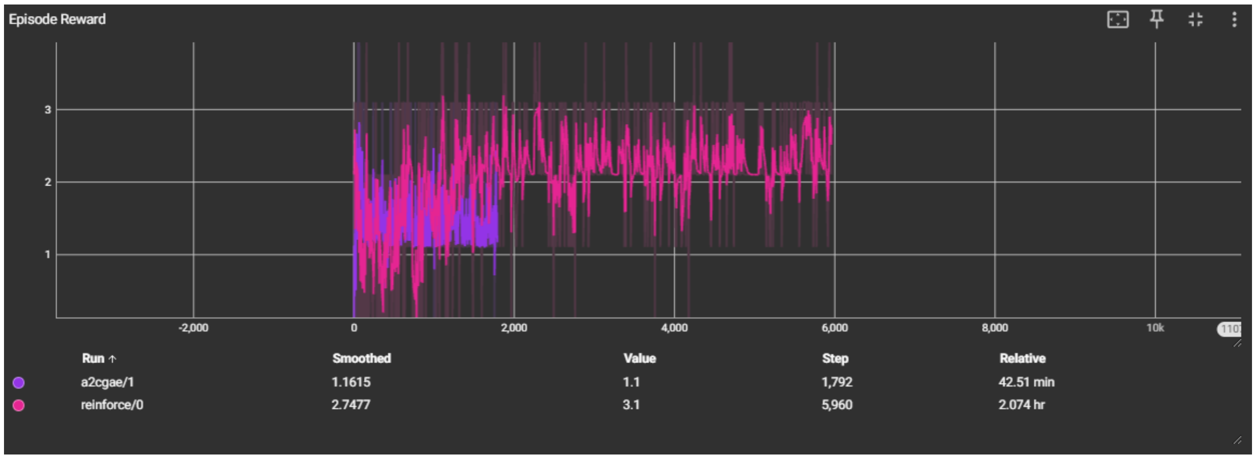
Rainbow DQN shows promising results in the first 200,000 steps, outperforming all other agents during this initial phase. Although it ends with a lower reward of around 3.1 due to an early training cutoff, this early lead suggests it could potentially reach higher rewards if given more training time.

DQN and **DDQN** use the temporal difference loss maintain a low loss during training.

Categorical DQN uses a distributional loss over discrete atoms representing the return distribution. Its gradually declining loss indicates a slower, ongoing refinement of this distributional estimate, potentially reflecting the added complexity of learning a return distribution rather than a single value.

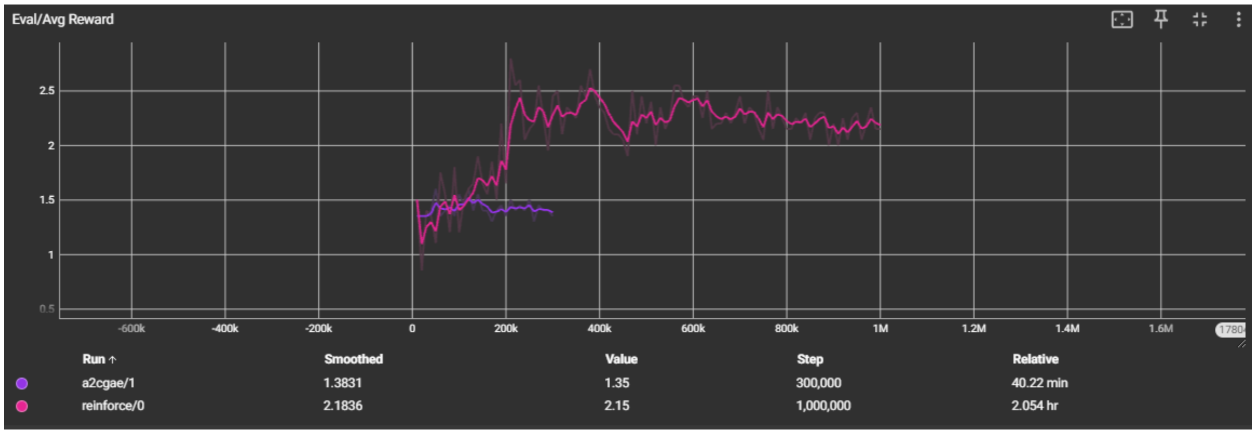
Rainbow DQN variants incorporate multiple components, which lead to higher initial loss and instability, as the models balance several objectives.

4. Policy-Based Methods Comparison



	Run ↑	Smoothed	Value	Step	Relative
●	a2cgae/1	1.1615	1.1	1,792	42.51 min
●	reinforce/0	2.7477	3.1	5,960	2.074 hr

Episode rewards across all models



	Run ↑	Smoothed	Value	Step	Relative
●	a2cgae/1	1.3831	1.35	300,000	40.22 min
●	reinforce/0	2.1836	2.15	1,000,000	2.054 hr

Evaluation average rewards across all models

The REINFORCE model initially shows promising learning progress, quickly increasing its average reward. However, it seems to reach a saddle point where further learning stalls, stabilizing its reward within a range of 2 to 2.5. This plateau suggests that REINFORCE may be getting stuck, possibly due to the limitations of the Monte Carlo approach.

On the other hand, the A2C-GAE agent appears unable to effectively learn in this environment, as its reward remains mostly stagnant over time without any noticeable improvement. This lack of progress indicates that A2C-GAE might struggle with learning an effective policy in this particular setup.