Scalability and Noise Resilience in Q-Transformer

Jun Wang
22320225@hdu.edu.cn

Abstract

The integration of deep reinforcement learning with Transformer architectures [Vaswani *et al.*, 2017] has demonstrated potential in addressing complex tasks under partial observability. However, challenges related to scalability and training efficiency remain. This paper investigates scalability through the Distributed Q-Transformer (DQT) framework, which combines the long-range dependency modeling of Transformers with the sample efficiency of Q-learning. By leveraging distributed training, DQT enables multiple agents to explore diverse trajectories in parallel, accelerating convergence in model-based environments.

Beyond scalability, we investigate how noise impacts the performance of Q-Transformer [qtr, 2023] and propose robust estimation techniques, including median-based filtering, to enhance its resilience. We systematically analyze the effects of noise injection and evaluate the robustness of the Q-Transformer within the MetaWorld benchmark [Yu and others, 2020]. Furthermore, we examine how different learning rate strategies influence performance under noisy conditions.

This work contributes to bridging the gap between Transformer-based policy representation and distributed reinforcement learning while addressing scalability and noise robustness, providing a viable solution for real-world robotic applications.

1 Introduction

Reinforcement learning (RL) enables agents to learn optimal policies through interactions with their environment. Traditional Q-learning methods estimate the expected cumulative reward for state-action pairs, facilitating the development of effective policies. However, these methods often encounter challenges when applied to high-dimensional action spaces and long-horizon planning due to their reliance on discrete action representations and limited capacity for modeling complex temporal dependencies.

Recent advancements have introduced Transformer architectures into RL, capitalizing on their proficiency in sequence modeling and capturing long-range dependencies. The Q-Transformer (QT)[qtr, 2023] model exemplifies this integration by employing an autoregressive approach to Q-learning, treating each action dimension as a separate time step. This methodology enhances the scalability and efficiency of Q-learning in continuous action spaces.

Despite these advancements, scaling RL algorithms to large, complex environments remains a significant challenge. Distributed Q-learning approaches have been proposed to address this issue by parallelizing the learning process across multiple agents or computational units. The Distributed Q-Transformer (DQT) extends the QT model into a distributed framework, enabling efficient learning in large-scale settings without compromising the benefits of autoregressive Q-learning. Another key challenge in real-world RL deployments is the presence of noise and uncertainty in the environment. We investigate the noise resilience of the Q-Transformer by studying how injected noise in the reward signal affects its performance. To mitigate adverse effects, a median-based filtering approach is applied to the reward signal to reduce the influence of outlier noise. This robust estimation leverages the median's resistance to extreme values, providing a more stable reward target under noisy conditions.

2 Related Work

2.1 Distributed Q-Learning

Q-learning is a model-free reinforcement learning algorithm that teaches an agent to assign values to each action it might take, conditioned on the agent being in a particular state. It does not require a model of the environment and can handle problems with stochastic transitions and rewards without requiring adaptations.

Distributed Q-learning approaches have been proposed to address the challenges of scaling RL algorithms to large, complex environments. These methods parallelize the learning process across multiple agents, enabling efficient learning in large-scale settings without compromising the benefits of Q-learning. [Ong *et al.*, 2015] discusses the adaptation of the DistBelief software framework[Dean *et al.*, 2012a] to efficiently train reinforcement learning agents.

2.2 Q-Transformer Architectures

The integration of Transformer models with Q-learning has led to the development of Q-Transformer architectures. By discretizing each action dimension and representing the Q-value of each action dimension as separate tokens and training on large offline datasets[qtr, 2023], Google DeepMind has proved q-transformer's feasibility and superiority in RL tasks. Furthermore, QT-TDM[Kotb *et al.*, 2025] integrates the robust predictive capabilities of Transformers as dynamics models(TDM)[Schubert *et al.*, 2023][Micheli *et al.*, 2023] with the efficacy of a model-free Q-Transformer to mitigate the computational burden associated with real-time planning,which achieves great success.

2.3 Noise Resilience in Reinforcement Learning

RL algorithms are often sensitive to noise in observations and reward signals, which can lead to unstable training or suboptimal policies. This has spurred research in robust reinforcement learning[Amarnath and Chatterjee, 2023], [Sun *et al.*, 2025], focusing on techniques to improve an agent's performance under noisy or uncertain conditions. Using a Huber loss for the Q-learning temporal-difference error (as done in DQT) is a common practice to reduce sensitivity to outlier rewards or errors, effectively improving stability. Domain randomization is another strategy wherein noise and perturbations are injected during training so that the learned policy generalizes better to variability in the environment. These approaches help ensure that the learned value function or policy does not overfit to idealized conditions and can tolerate random disturbances.

3 Background

Reinforcement learning (RL) is a paradigm where agents learn to make decisions by interacting with an environment, aiming to maximize cumulative rewards. The process is formalized as a Markov Decision Process (MDP), defined by a tuple (S, A, R, T, γ), where S represents the state space, A the action space, R the reward function, T the transition function, and γ the discount factor. The objective is to learn a policy $\pi : S \to A$ that maximizes the expected cumulative reward.

Q-Learning is a foundational model-free RL algorithm that estimates the optimal action-value function $Q^*(s, a)$, representing the maximum expected return achievable from state s and action a. Traditional Q-learning methods, such as Deep Q-Networks (DQN), utilize neural networks to approximate $Q^*(s, a)$ for discrete action spaces. However, these methods face challenges when applied to continuous and high-dimensional action spaces due to the need for discretization, which can lead to inefficiencies and scalability issues.

Autoregressive Q-Learning addresses these challenges by treating each action dimension as a separate time step, allowing for more efficient prediction of Q-values in continuous action spaces. The Q-Transformer (QT) model employs this approach, utilizing Transformer architectures to model the dependencies between action dimensions. This method enhances the scalability and performance of Q-learning in high-dimensional action spaces by leveraging the Transformer's ability to capture complex temporal dependencies.

Despite the advancements offered by QT, scaling these models to large, complex environments remains a significant challenge. Distributed Q-Learning approaches have been proposed to address this issue by parallelizing the learning process across multiple agents or computational units. The Distributed Q-Transformer (DQT) extends the QT model into a distributed framework, enabling efficient learning in large-scale settings. By leveraging the parallel processing capabilities of Transformer architectures, DQT enhances scalability and sample efficiency, making it suitable for real-time applications in complex environments.

Noise and uncertainty in the environment present another practical challenge for RL algorithms. Real-world robotic systems, for instance, often have to deal with sensor noise, delayed or missing observations, and stochastic disturbances in the reward signal. If the reward signal is corrupted by random noise, a Q-learning algorithm may receive misleading feedback, overestimating or underestimating the true value of certain actions, which can slow convergence or lead to suboptimal policies. Therefore, it is important for algorithms like Q-Transformer to be resilient to noisy rewards and observations. Techniques from robust statistics and control can be employed to improve noise tolerance. One simple approach is to aggregate multiple noisy samples and use a robust statistic (such as the median) as the estimate of the true reward.

4 Methodology

The flow and architecture of DQT are illustrated in Fig. 1. This section details the interaction between agents and the central server. And the way we used to test and improve the robust of QT.

4.1 DQT

Action Selection

The agent selects action a_t using an ϵ-greedy policy:

$$a_t = \begin{cases} \text{Random action}, & \text{if } \epsilon > \text{random.Uniform}(0, 1) \\ \arg\max_a Q(s_t, a; \theta), & \text{otherwise} \end{cases} \tag{1}$$

where ϵ decays over time:

$$\epsilon_t = \max\left(\epsilon_{\min}, \epsilon_{\max} \times \left(1 - \frac{t}{T_{\text{decay}}}\right)\right) \tag{2}$$

Experience Collection

After executing a_t, the agent collects experience:

$$(s_t, a_t, r_t, s_{t+1}) \tag{3}$$

and stores it in the replay buffer $\mathcal{D}$:

$$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\} \tag{4}$$

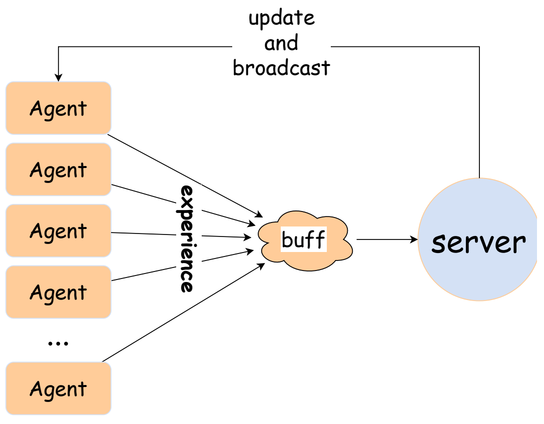
Server Updates Q-Network

The server samples a batch from multiple agents and computes the TD target:

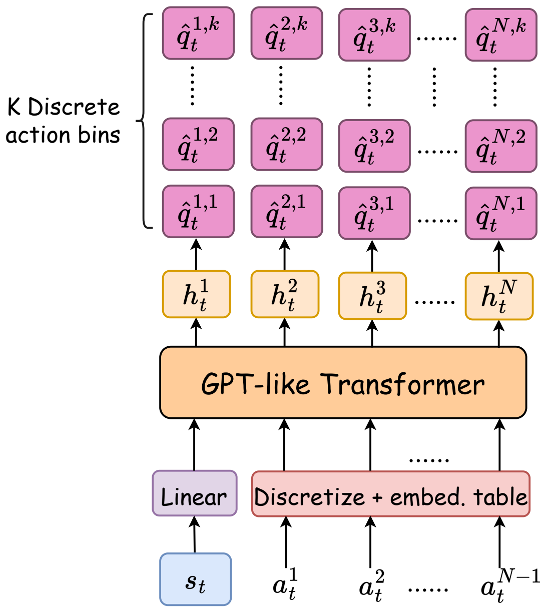
$$y_t = r_t + \gamma \max_{a'} Q_{\text{target}}(s_{t+1}, a'; \theta^-) \tag{5}$$

TD loss is then computed as:

$$\mathcal{L}_{\text{TD}} = E_{(s,a,r,s')\sim\mathcal{D}}\left[\left(y_t - Q(s_t, a_t; \theta)\right)^2\right] \tag{6}$$



(a) The whole flow


K Discrete action bins
$\hat{q}_t^{1,k}$ $\hat{q}_t^{2,k}$ $\hat{q}_t^{3,k}$ ...... $\hat{q}_t^{N,k}$
$\hat{q}_t^{1,2}$ $\hat{q}_t^{2,2}$ $\hat{q}_t^{3,2}$ ...... $\hat{q}_t^{N,2}$
$\hat{q}_t^{1,1}$ $\hat{q}_t^{2,1}$ $\hat{q}_t^{3,1}$ ...... $\hat{q}_t^{N,1}$
h_t^1 h_t^2 h_t^3 ...... h_t^N
GPT-like Transformer
Linear
Discretize + embed. table
s_t a_t^1 a_t^2 ...... a_t^{N-1}

(b) Q-Transformer

Figure 1: (a) the overall DQT flow, where multiple agents interact with the environment in parallel and send experience to a central server, and (b) the Q-Transformer architecture (reproduced from [Kotb *et al.*, 2025]), which tokenizes the state and discretizes action dimensions for input to a GPT-style Transformer.

An additional conservative Q-learning loss is applied:

$$\mathcal{L}_{\text{CQL}} = E_{s\sim\mathcal{D}}\left[\log\sum_a \exp Q(s,a) - E_{a\sim\mathcal{D}}Q(s,a)\right] \quad (7)$$

Final optimization objective:

$$\mathcal{L} = \mathcal{L}_{\text{TD}} + \eta\mathcal{L}_{\text{CQL}} \quad (8)$$

To train the Q-funtion,a per-dimension Bellman update is defined as follows:

$$Q(s_t, a_t^{1:i-1}, a_t^i) \leftarrow \begin{cases} \max\limits_{a_t^{i+1}} Q(s_t, a_t^{1:i}, a_t^{i+1}) & \text{if } i < N \\ r_t + \gamma \max\limits_{a_{t+1}^1} Q(s_{t+1}, a_{t+1}^1) & \text{if } i = N. \end{cases} \quad (9)$$

Target Network Update
The target Q-network is updated using an exponential moving average (EMA):

$$\theta^- \leftarrow \tau\theta + (1-\tau)\theta^- \quad (10)$$

Agent Updates Q-Network
Agents receive the updated Q-network from the server:

$$\theta \leftarrow \theta_{\text{server}} \quad (11)$$

4.2 Learning rate Scheduling

linear decay with logn

$$\eta_{\text{lr}}(t) = \begin{cases} \max\left(1 - \dfrac{t}{T},\ 0\right), & n = 1 \\ \max\left(1 - \dfrac{\ln(n)\cdot t}{T},\ 3\times 10^{-5}\right), & n > 1 \end{cases}$$

log curve decay

$$\eta_{\text{lr}}(t) = \begin{cases} 1 - \dfrac{t}{T}, & n = 1 \\ \dfrac{1}{1 + \ln(n)\cdot\dfrac{t}{T}}, & n > 1 \end{cases}$$

linear decay with n

$$\eta_{\text{lr}}(t) = \max\left(1 - \frac{n\cdot t}{T},\ 0\right)$$

linear decay

$$\eta_{\text{lr}}(t) = 1 - \frac{t}{T}$$

4.3 Evaluating the Robustness of QT

Here we evaluate the Robustness of QT by adding noise to the reward. And to combat the negative impact of stochastic noise on reward signals, we generate multiple noisy samples and use their median as a robust reward estimate.

add noise

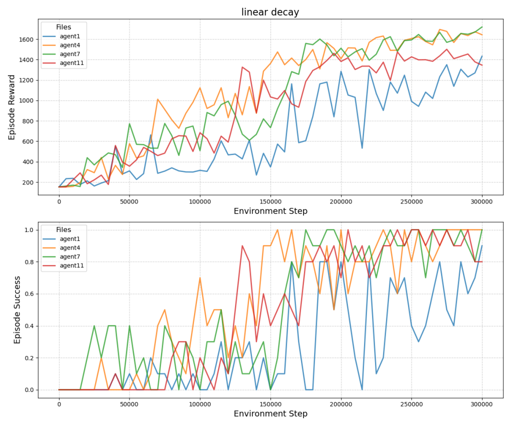
$$noise \sim N(\mu, \sigma^2) \quad (12)$$

$$r_t = r_t + noise \quad (13)$$

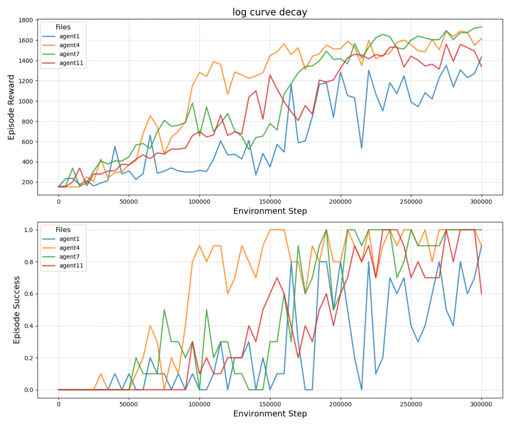
use robust reward

$$noise_i \sim N(\mu, \sigma^2), i = 1, 2, \ldots, 100 \quad (14)$$

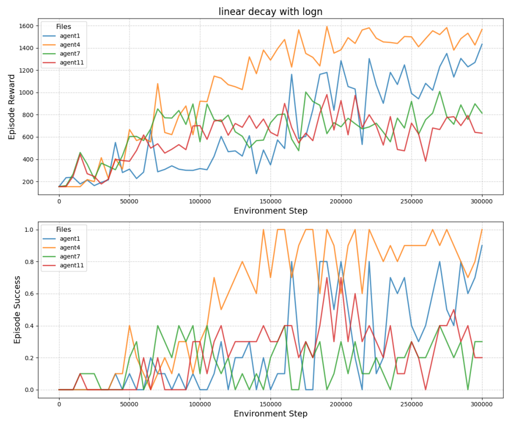
$$r_{robust} = median(\{r + noise_i\}_{i=1}^{100}) \quad (15)$$



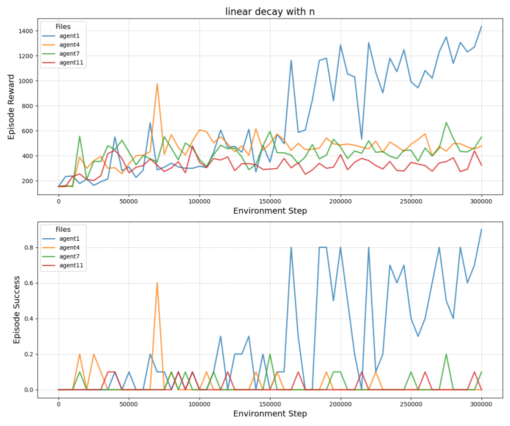
(a) linear decay



(b) log curve decay



(c) linear decay with logn



(d) linear decay with n

Figure 2: a,b,c are the results of four different methods to deploy learning rate, as the title implies. The top picture is reward vs step, and below is success vs step. The legend represents different agent numbers.

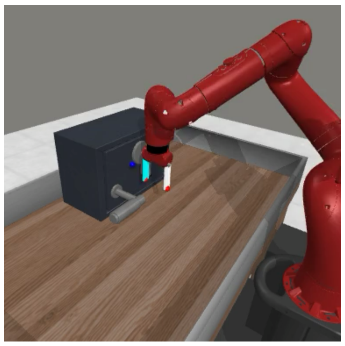


Figure 3: door-unlock

5 Experiments

5.1 Description and Details

Benchmarks

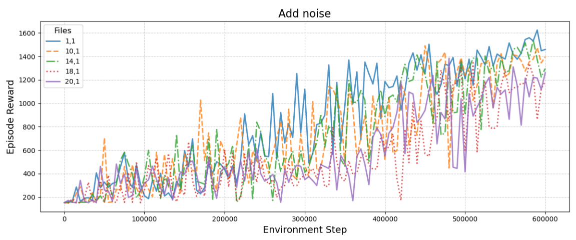
Limited by the lack of computing resources(1 NVIDIA GeForce RTX 3060), we just evaluate our DQT model in a robotic manipulation task named door-unlock Fig. 3 from Metaworld[Yu and others, 2020].

DQT Experimental setup

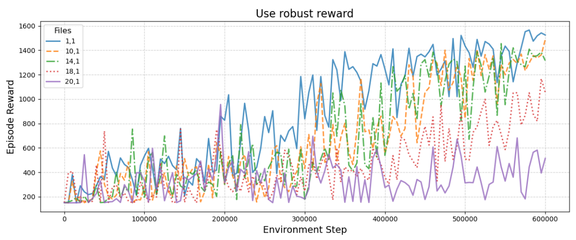
We use four different ways 4.2 to change our learning rate deploy when updating the server to find out which one has the best result, t is current step, T is total training step = 300000, n is the number of agents $n \in \{1, 4, 7, 11\}$. We set 6 individual experiments and average them to get the results.

QT robust Experiment setting

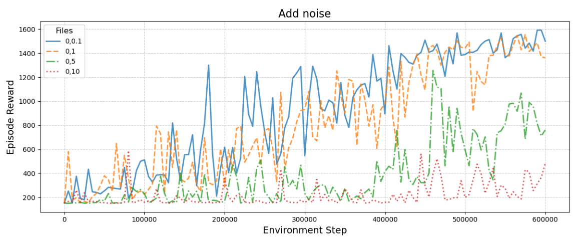
To find out at what level of noise will QT fail and demonstrate the different influence between μ and σ. We set up two controlled experiments:1) with fixed $\mu = 0$ and $\sigma \in \{0.1, 1, 5, 10\}$. 2) with fixed $\sigma = 1$ and $\sigma \in \{1, 10, 14, 18, 20\}$. We also try to enhance QT's robustness with a method shown in 4.3, the experiments are similar to the noise one except for the robust reward part. And prove that method is beneficial for other tasks, shown in Fig. 5



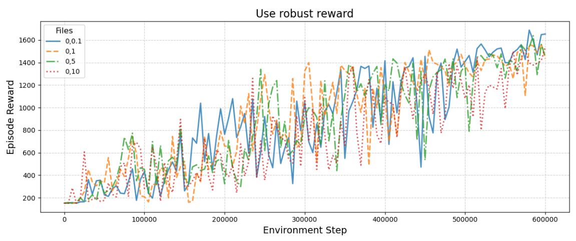
(a) add noise with fixed variance



(b) robust reward with fixed variance



(c) add noise with fixed mean



(d) robust reward with fixed mean

Figure 4: The legend represents the specific value of mean and variance, mean is on the left and variance is on the right. The whole left side of the picture is the result of adding noise and the other side uses robust reward.

6 Experimental Results

6.1 Effect of Learning Rate Schedules in DQT

Experiment Setup

We investigate how different learning rate decay strategies affect the performance of Distributed Q-Transformer (DQT) under varying numbers of agents. Specifically, we compare four strategies: linear decay, logarithmic decay (log curve), linear decay scaled by $\log(n)$, and linear decay scaled by n, where n is the number of agents.

Observations

As shown in Fig. 2, DQT with linear decay scaled by n fails to converge under all tested configurations. Linear decay with $\log(n)$ performs slightly better but breaks down as the number of agents increases beyond 7. In contrast, plain linear decay and log curve decay yield consistent performance across different agent counts.

Analysis

These results suggest that over-aggressive decay scaling strategies (especially those directly proportional to n) can lead to premature learning rate reduction, harming convergence. Furthermore, our synchronized update mechanism may introduce gradient variance that is not well-compensated by naïvely scaled learning rates. This highlights the need for more principled adaptive learning rate mechanisms when scaling to larger agent populations.

6.2 Robustness of Q-Transformer under Noisy Rewards

Experiment Setup

To test the robustness of the Q-Transformer (QT), we introduce Gaussian noise to the reward signal and compare standard QT with a version using median-based reward smoothing. We examine the effects of noise with varying mean (μ) and variance (σ^2), as shown in Fig. 4.

Observations

Without robust smoothing, QT exhibits significant degradation under both high-mean and high-variance noise. When median filtering is applied, the model becomes less sensitive to noise variance, but remains sensitive to changes in noise mean.

Analysis

This behavior is expected: the median is a robust estimator of central tendency but remains affected by consistent shifts in the mean. These findings suggest that while median-based reward smoothing provides partial robustness, it does not fully resolve the problem. More advanced techniques—such as explicit noise modeling or adaptive reward normalization—may be needed to improve QT's resilience in real-world scenarios.

7 Conclusion and Future Work

This work explored the scalability and robustness of Q-Transformer through the Distributed Q-Transformer (DQT) framework and reward smoothing techniques.

7.1 Limitations of Current Design

Although DQT enables parallel exploration, our current implementation with synchronous updates and fixed learning rate decay strategies failed to deliver significant performance improvements. We attribute this to:

- **Synchronization Bottlenecks**: Parameter updates occur only after each full episode, leading to latency and stale gradients.

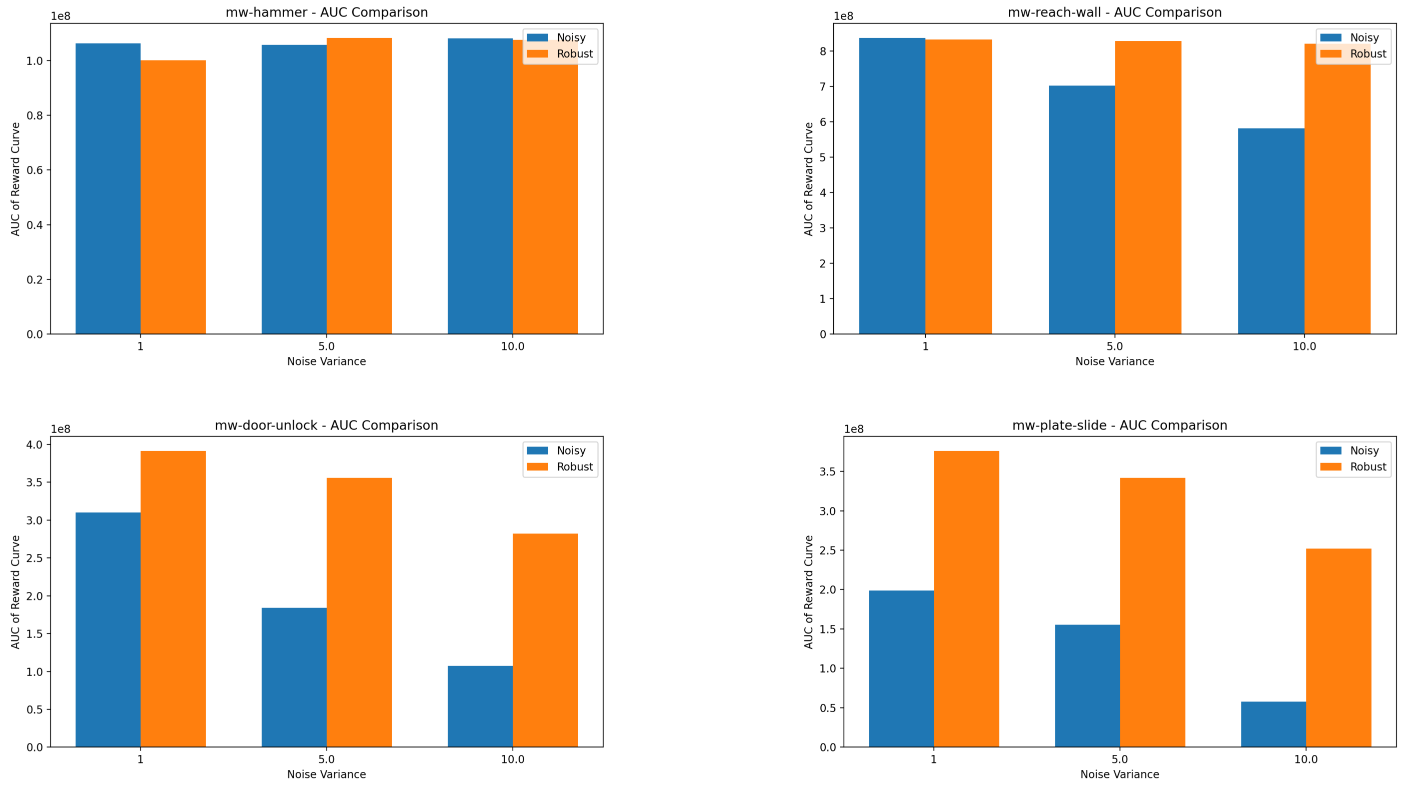


Figure 5: Comparison of AUC (the area between episode reward and environment step-600000)values between noisy and robust reward settings across four different tasks. AUC is calculated as the integral of success rate over training steps, reflecting both the learning speed and stability of the agent. The left side of each subplot shows the results under noisy reward conditions, while the right side shows results using robust rewards. The legends indicate the specific values of mean (left) and variance (right) for each method. The four tasks shown are: (1) Hammer, (2) Reach Wall, (3) Door Unlock, and (4) Plate Slide.

- **Homogeneous Experience**: Agents running identical environments may collect similar experiences, reducing diversity and limiting learning benefits.
- **Inflexible Scheduling**: Naively scaling the learning rate based on agent count introduces instability rather than convergence gains.

7.2 Robustness to Reward Noise

Our experiments confirmed that QT is sensitive to noisy reward signals. The use of median filtering mitigates the impact of outliers and high-variance noise, but remains insufficient against biased noise (non-zero mean). This suggests that while simple statistical filters help, they are not a complete solution.

7.3 Future Directions

To address the above limitations, future work will explore:

- **Asynchronous Architectures**: Inspired by DistBelief's Downpour SGD [Dean *et al.*, 2012b], asynchronous gradient updates may alleviate synchronization overhead and improve learning stability.
- **Diverse Experience Generation**: Techniques such as domain randomization or environment augmentation can increase data diversity in multi-agent settings.
- **Advanced Noise-Handling**: Beyond median smoothing, we plan to investigate adaptive reward shaping, noise-aware Q-value estimation, and robust loss formulations (e.g., Huber loss with adaptive thresholds).

By improving both the scalability and robustness of the Q-Transformer, we aim to make it more suitable for real-world deployment in complex, noisy environments.

References

[Amarnath and Chatterjee, 2023] Chandramouli Amarnath and Abhijit Chatterjee. A novel approach to error resilience in online reinforcement learning. In *2023 IEEE 29th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pages 1–7. IEEE, 2023.

[Dean *et al.*, 2012a] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc' aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Ng. Large scale distributed deep networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information*

Processing Systems, volume 25. Curran Associates, Inc., 2012.

[Dean *et al.*, 2012b] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *NIPS*, 2012.

[Kotb *et al.*, 2025] Mostafa Kotb, Cornelius Weber, Muhammad Burhan Hafez, and Stefan Wermter. Qt-tdm: Planning with transformer dynamics model and autoregressive q-learning. *IEEE Robotics and Automation Letters*, 10(1):112–119, 2025.

[Micheli *et al.*, 2023] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models, 2023.

[Ong *et al.*, 2015] Hao Yi Ong, Kevin Chavez, and Augustus Hong. Distributed deep q-learning, 2015.

[qtr, 2023] Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *7th Annual Conference on Robot Learning*, 2023.

[Schubert *et al.*, 2023] Ingmar Schubert, Jingwei Zhang, Jake Bruce, Sarah Bechtle, Emilio Parisotto, Martin Riedmiller, Jost Tobias Springenberg, Arunkumar Byravan, Leonard Hasenclever, and Nicolas Heess. A generalist dynamics model for control, 2023.

[Sun *et al.*, 2025] Chenglu Sun, Shuo Shen, Wenzhi Tao, Deyi Xue, and Zixia Zhou. Noise-resilient symbolic regression with dynamic gating reinforcement learning, 2025.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[Yu and others, 2020] Tianhe Yu et al. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. pages 1094–1100, 2020.