Deep Reinforcement Learning-based Mobile Robot Navigation Using Truncated Quantile Critics

Sang Uk Bae
School of Electronic and Electrical Engineering
Kyungpook National University
Daegu, Republic of Korea
qlfekd12@knu.ac.kr

Dong Seog Han
School of Electronic and Electrical Engineering
Kyungpook National University
Daegu, Republic of Korea
dshan@knu.ac.kr

***Abstract*—Navigating cluttered indoor environments requires robust continuous control under sensor uncertainty. This paper presents a deep reinforcement learning framework using truncated quantile critics (TQC) to learn velocity control directly from LiDAR data. Raw 3D point clouds from a simulated Ouster LiDAR are projected to 2D scans and discretized into sectors to form the state representation. To ensure robustness, we train a skid-steer robot in a randomized ROS 2–Gazebo environment with dynamic start–goal and obstacle configurations. A composite reward function utilizing zone-based collision detection is designed to encourage safe and smooth trajectories. Experimental results on a 16×16 m map demonstrate that the TQC-based policy achieves high success rates with low collision frequency, demonstrating the efficacy of distributional reinforcement learning for reliable mobile robot navigation.**

***Index Terms*—Deep reinforcement learning, Mobile robot navigation, Truncated quantile critics, LiDAR, ROS2, Autonomous driving**

I. Introduction

Autonomous mobile robots are increasingly deployed in factories, warehouses, and other indoor logistics environments to automate material handling and transport. These environments contain narrow aisles, dense static structures, and dynamic variations, requiring a robot to reliably reach goal locations from arbitrary poses while avoiding collisions and generating smooth, feasible motions. Classical navigation stacks typically combine a global planner with a local planner such as the dynamic window approach (DWA) [1], timed-elastic-band (TEB) [2], or A* variants. Although these methods are mature, they often depend on simplified motion models, carefully tuned cost weights, and accurate maps. Their performance can degrade in cluttered, partially observed scenes or when perception, planning, and control modules are not perfectly synchronized.

Deep reinforcement learning (DRL) offers an end-to-end alternative where a navigation policy is learned directly from interaction [3]. With rich sensor inputs and a suitable reward function, a DRL agent can map observations to continuous control commands that jointly account for goal reaching, obstacle avoidance, and motion smoothness. However, applying DRL to mobile robotics faces key challenges: ensuring stable learning in continuous action spaces, designing rewards that balance safety and efficiency, and constructing realistic simulation pipelines that reflect real-world dynamics.

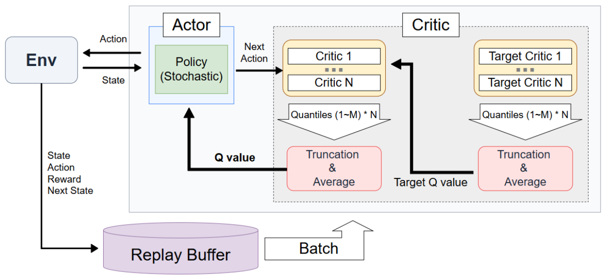


Fig. 1: Schematic of the TQC-based learning architecture. The system employs an ensemble of critics and a truncation mechanism to provide stable value estimates for the actor.

In this work, we formulate indoor navigation as a continuous-control Markov decision process in a ROS 2 [4]–Gazebo [5] simulation using a skid-steer robot model equipped with a LiDAR sensor. The raw LiDAR data is processed into a sector-based representation and concatenated with the relative goal distance, heading error, and the previous action vector. At each time step, the DRL agent outputs linear and angular velocities. The control objective is to reach a goal region within a fixed horizon while avoiding collisions with randomly placed obstacles and minimizing oscillatory movements.

To address this problem, we adopt the truncated quantile critics (TQC) algorithm [6], a distributional actor–critic method that maintains multiple quantile critics to control overestimation bias. The environment implements a composite reward function that includes: (i) progress towards the goal, (ii) a curvature penalty to suppress abrupt rotations, (iii) zone-based safety penalties utilizing sector-specific thresholds, and (iv) a time-step penalty to encourage efficiency.

The overall learning architecture is illustrated in Fig. 1. The agent interacts with the simulation environment, storing state transitions in a replay buffer. The TQC algorithm utilizes an ensemble of critics to estimate the return distribution. To mitigates overestimation, the top quantiles from these critics are truncated before averaging, yielding a robust Q-value estimate that guides the stochastic policy update.

The main contributions of this paper are summarized as follows:

- We design a ROS 2–Gazebo training environment that

facilitates robust learning through domain randomization [7], varying the poses of the robot, goal, and obstacles in every episode to prevent overfitting.
- We propose a comprehensive reward function that balances goal-directed behavior with safety, specifically employing a zone-based collision detection mechanism to refine obstacle avoidance in critical sectors.
- We successfully adapt the TQC algorithm for the mobile robot navigation task, demonstrating that its distributional nature contributes to stable convergence and high success rates in cluttered environments.
- We empirically evaluate the learned policies, reporting metrics on success rate, collision frequency, and path efficiency, confirming the validity of the proposed framework.

II. Related Work

A. Classical vs. Learning-Based Navigation

Classical navigation pipelines typically separate localization, mapping, and planning. While local planners like DWA and TEB are computationally efficient, they struggle in complex environments due to their reliance on hand-crafted cost functions and simplified kinematics. DRL-based approaches, conversely, learn collision avoidance and path planning end-to-end. Early works utilized value-based methods (*e.g.*, deep Q-network (DQN) [8]) with discrete actions, while recent studies employ continuous actor-critic algorithms (*e.g.*, deep deterministic policy gradient (DDPG) [9], soft actor-critic (SAC) [10]). However, these methods often suffer from training instability and value overestimation when facing high-dimensional sensor data [11].

B. Distributional RL and Truncated Quantile Critics

Distributional reinforcement learning models the full return distribution instead of only its expectation. Categorical and quantile-based methods (*e.g.*, C51, QR-DQN, IQN [12], FQF) have shown improved stability and sample efficiency by better capturing uncertainty in returns. For continuous control, quantile critics have been integrated with actor–critic methods to represent the action-value distribution with multiple quantile estimates [13].

TQC [6] extend this line of work by sorting all critic quantiles and explicitly discarding the highest ones, thereby controlling overestimation bias and improving the robustness of policy evaluation in continuous-action settings. Combined with entropy-regularized policies (as in SAC [10]), TQC has been reported to achieve strong performance on standard continuous-control benchmarks by stabilizing training and yielding more conservative value estimates in risky states.

C. Positioning of This Work

Unlike prior works relying on simplified sensors or static maps, we integrate TQC into a high-fidelity ROS 2 pipeline. Our approach distinguishes itself by: (i) Using TQC to stabilize learning in cluttered environments. (ii) Implementing a realistic perception frontend that converts simulated Ouster 3D LiDAR point clouds to 2D laser scans, bridging the gap to real-world hardware. (iii) Employing a zone-based reward structure that explicitly reflects robot kinematics for enhanced safety. This combination allows for systematic evaluation of distributional RL as a robust backbone for autonomous navigation.

III. Problem Formulation and System Setup

A. Robot Platform and Simulation Environment

We simulate a four-wheeled skid-steer mobile robot (parametrized after the *Bunker* platform) in a ROS 2 [4] and Gazebo [5] environment. The robot is controlled via linear velocity $v_t \in [-1.7, 1.7]\,\mathrm{m/s}$ and angular velocity $\omega_t \in [-3.14, 3.14]\,\mathrm{rad/s}$. The training environment consists of a $16 \times 16\,\mathrm{m}$ arena with 15 cylindrical obstacles. To ensure policy robustness, the poses of the robot, goal, and obstacles are randomly reset at the start of each episode, leveraging domain randomization techniques [7]. The control frequency is set to $10\,\mathrm{Hz}$.

Perception relies on a simulated 3D Ouster LiDAR, which is converted into a 2D laser scan and compressed into an $N_L = 80$ dimensional range vector by taking the minimum distance in each sector. Episodes terminate upon reaching the goal (success), colliding with an obstacle (failure), or exceeding 500 time steps.

B. Markov Decision Process

We model the navigation task as a Markov decision process (MDP)

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma), \tag{1}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, P the transition kernel, R the reward function, and $\gamma \in (0, 1)$ the discount factor. We set $\gamma = 0.99$ in all experiments.

At time t, the state $s_t \in \mathcal{S}$ concatenates LiDAR and agent-level features:

$$s_t = \left[\, d_t,\ d_t^{\text{goal}},\ \theta_t^{\text{err}},\ \tilde{v}_{t-1},\ \tilde{\omega}_{t-1} \,\right] \in \mathbb{R}^{N_L+4}, \tag{2}$$

where $d_t \in \mathbb{R}^{N_L}$ is the sector-wise minimum LiDAR range vector (clipped to $[0, d_{\max}]$), d_t^{goal} is the distance to the goal, θ_t^{err} the heading error, and $(\tilde{v}_{t-1}, \tilde{\omega}_{t-1})$ the previous linear and angular commands normalized by $(v_{\max}, \omega_{\max})$.

The policy outputs a normalized action $\tilde{a}_t = [\tilde{v}_t, \tilde{\omega}_t]^\top \in [-1, 1]^2$, which is mapped to physical commands by

$$v_t = v_{\max}\,\tilde{v}_t, \qquad \omega_t = \omega_{\max}\,\tilde{\omega}_t. \tag{3}$$

These commands are sent as a `geometry_msgs/Twist` to the simulator and applied for one control interval Δt.

C. Reward Design

The reward function r_t is designed to encourage efficient path planning while strictly enforcing safety. It is composed of a sparse terminal reward and dense shaping rewards:

$$r_t = r_t^{\text{term}} + r_t^{\text{nav}} + r_t^{\text{safety}}. \tag{4}$$

TABLE I: Key Environment and Reward Parameters

Parameter	Value
Map size / Obstacles	$16 \times 16\,\mathrm{m}$ / 15
Action Limits (v, ω)	$\pm1.7\,\mathrm{m/s}, \pm3.14\,\mathrm{rad/s}$
LiDAR Sectors (N_L) / Zones (N_Z)	80 / 8
Goal Threshold δ_{goal}	$0.42\,\mathrm{m}$
Safety Margin ρ	1.5

a) Terminal and Navigation Rewards

A large reward $(+10)$ is granted for reaching the goal, while a penalty (-10) is applied for collisions. The navigation term r_t^{nav} includes a progress reward proportional to the decrease in distance to the goal and a heading reward for facing the target. A small time penalty is subtracted at each step to encourage faster completion.

b) Zone-based Safety Penalty (r_t^{safety})

To reflect the robot's kinematics and geometry, we employ a *zone-based* collision detection scheme rather than a simple minimum-distance check. The LiDAR field-of-view is partitioned into $N_Z = 8$ angular zones. For each zone i, we define a distance deficit $\delta_t^{(i)}$ if the obstacle distance $z_t^{(i)}$ falls below a safety threshold $d_{\text{thr}}^{(i)}$:

$$r_t^{\text{safety}} = -w_{\text{obs}} \sum_{i=1}^{N_Z} w^{(i)} \cdot \max\left(0, 1 - \frac{z_t^{(i)}}{\rho \cdot d_{\text{thr}}^{(i)}}\right), \quad (5)$$

where ρ is a safety margin factor and $w^{(i)}$ represents the importance weight of each zone (*e.g.*, frontal zones have higher weights). This formulation allows the agent to learn precise maneuvering in tight spaces by receiving granular feedback on which part of the robot is at risk. Finally, an additional curvature penalty is applied to discourage excessive in-place rotation, and the total reward is clipped to $[-1, 1]$ for training stability.

IV. TQC-based Navigation Policy

A. Overview of Truncated Quantile Critics

We adopt truncated quantile critics (TQC) [6] to handle the continuous action space of the mobile robot. TQC extends distributional reinforcement learning by approximating the return distribution $Z(s, a)$ using a set of quantiles, rather than a single scalar expectation. In our framework, we employ an ensemble of N_C critic networks, where each critic predicts N_Q quantiles.

To mitigate the overestimation bias often observed in Q-learning, TQC applies a truncation mechanism during the target value calculation. For a next state–action pair (s_{t+1}, a_{t+1}), the quantiles from all N_C critics are pooled and sorted. The top k estimates are discarded (truncation), and the target value y_t is computed by averaging the remaining quantiles in the set $\mathcal{Z}_{\text{trunc}}$:

$$y_t = r_t + \gamma \left(\frac{1}{|\mathcal{Z}_{\text{trunc}}|} \sum_{z \in \mathcal{Z}_{\text{trunc}}} z - \alpha \log \pi_\theta(a_{t+1} \mid s_{t+1}) \right). \quad (6)$$

Here, α is the temperature parameter balancing the entropy maximization, which is automatically tuned during training. The critics are updated to minimize the quantile Huber loss against this target.

B. Network Architecture

The policy and value functions are parameterized using standard feedforward neural networks.

Actor Network

The actor is a stochastic policy network parameterized as a multi-layer perceptron (MLP). It consists of three hidden layers with 256 units each, using ReLU activations. The network takes the state vector s_t as input and outputs the mean μ and log-standard deviation $\log \sigma$ of a diagonal Gaussian distribution. Actions are sampled using the reparameterization trick and bounded to the range $[-1, 1]$ via a $\tanh$ activation function. Finally, the actions are scaled to match the robot's physical velocity limits $(v_{\max}, \omega_{\max})$.

Critic Networks

The critic ensemble consists of $N_C = 5$ parallel networks. Each critic follows the same architecture: a three-layer MLP (256 units per layer) with ELU activations. The final layer outputs a vector of size $N_Q = 25$, representing the quantiles of the return distribution. A set of target critics with identical architecture is maintained and updated via Polyak averaging to stabilize the learning process.

Prioritized Replay Buffer

We utilize a prioritized replay buffer [14] to improve sample efficiency. Transitions with higher temporal-difference (TD) errors are sampled more frequently, allowing the agent to focus on unexpected or informative experiences.

C. Training Procedure

The training process follows an off-policy actor-critic loop. Transitions (s_t, a_t, r_t, s_{t+1}) collected from the Gazebo environment are stored in the replay buffer. A single training iteration consists of the following steps:

1) **Critic Update:** A batch of transitions is sampled. The target distribution is constructed using the truncation mechanism described in (6). The critics are updated by minimizing the quantile regression loss averaged over all valid quantile pairs.
2) **Actor Update:** The policy is updated by maximizing the expected Q-value (averaged over all critics and quantiles) combined with the entropy term.
3) **Entropy Tuning:** The temperature α is adjusted via gradient descent to maintain a target entropy, preventing premature convergence to a deterministic policy.
4) **Soft Update:** The target network parameters are updated using a moving average of the online network parameters.

TABLE II: Core TQC hyperparameters

Hyperparameter	Value
Discount factor γ	0.99
Batch size	256
Replay buffer size	10^6
Critics (N_C) / Quantiles (N_Q)	5 / 25
Dropped quantiles (k)	2
Target update τ	0.005
Learning rates (Actor/Critic/Ent)	3×10^{-4}

D. Implementation in ROS 2

The proposed framework is fully integrated into the ROS 2 [4] ecosystem to ensure compatibility with real-world robotic standards.

- **Environment Node:** This node wraps the Gazebo [5] simulation, handling the synchronization of physics steps and publishing sensor data (LiDAR, odometry). It implements the zone-based reward calculation and exposes standard ROS services for resetting episodes.
- **Training Node:** Implementing the TQC algorithm in PyTorch, this node interacts with the environment node asynchronously. It sends velocity commands and receives observations, managing the replay buffer and network updates on a dedicated GPU.
- **Evaluation Node:** A separate node is designed for deterministic policy evaluation. It bypasses the exploration noise and logs performance metrics such as success rate, collision frequency, and path efficiency for post-hoc analysis.

V. Experiments and Results

A. Experimental Setup

We evaluate the proposed navigation policy in the ROS 2 [4]–Gazebo [5] simulation environment. The robot operates in a $16 \times 16\,\mathrm{m}$ arena with 15 cylindrical obstacles. At the start of each episode, the poses of the robot, goal, and obstacles are uniformly randomized with a minimum separation of $2.5\,\mathrm{m}$ to ensure feasible initial configurations.

Training is performed off-policy with a control frequency of $10\,\mathrm{Hz}$. Transitions (s_t, a_t, r_t, s_{t+1}) are stored in a prioritized replay buffer. The training consists of 1,000,000 steps, starting with a 25,000-step warm-up phase. Evaluation rollouts are conducted every 5,000 steps using a deterministic policy to monitor convergence. Table II lists the key hyperparameters. We use an ensemble of $N_C = 5$ critics with $N_Q = 25$ quantiles each, discarding the top $k = 2$ quantiles to mitigate overestimation. The entropy coefficient α is automatically tuned.

B. Training Analysis

We analyze the training dynamics using metrics averaged over the final 10% of steps (Table III). The entropy coefficient α converges to ≈ 0.0186 with low variance, indicating a shift towards a deterministic policy. Both actor and critic losses stabilize with negligible drift, and the mean Q-values plateau around 14.4. The bounded maximum Q-values confirm that

TABLE III: Training metrics (Final 10%)

Metric	Mean / Median	Slope
Entropy α	0.0186	—
Actor loss	-14.36	$+8.9 \times 10^{-7}$
Critic loss	0.169	$+4.9 \times 10^{-8}$
Q values	14.39	-9.0×10^{-7}

TABLE IV: Navigation success rates over 6 independent runs

Test Run	**Success Rate** ($\geq 3.0\,\mathrm{m}$)	**Success Rate** ($\geq 6.0\,\mathrm{m}$)
1	0.99	0.97
2	0.98	0.97
3	**1.00**	0.98
4	0.99	0.97
5	**1.00**	0.96
6	**1.00**	0.97
Average	*0.993*	*0.970*

quantile truncation effectively controls overestimation in risky states.

Fig. 2 visualizes these trends. The stable plateau behavior across all metrics demonstrates robust convergence of the TQC algorithm.

C. Navigation Performance

To rigorously evaluate the policy's robustness and consistency, we conducted 6 independent test runs. Each run consisted of randomized episodes, separated into two difficulty levels based on the start–goal distance: *Medium* ($\geq 3.0\,\mathrm{m}$) and *Long* ($\geq 6.0\,\mathrm{m}$).

Table IV details the success rates for each run. In the Medium-distance scenarios, the policy demonstrated near-perfect performance, with success rates ranging from 0.98 to 1.0. In the more challenging Long-distance scenarios, which require navigating through denser obstacle fields, the policy maintained high stability, with success rates fluctuating slightly between 0.96 and 0.98. Across all 6 runs, the average success rates were approximately 99.3% for medium distances and 97.0% for long distances. These results confirm that the proposed TQC framework yields a highly reproducible and reliable navigation policy.

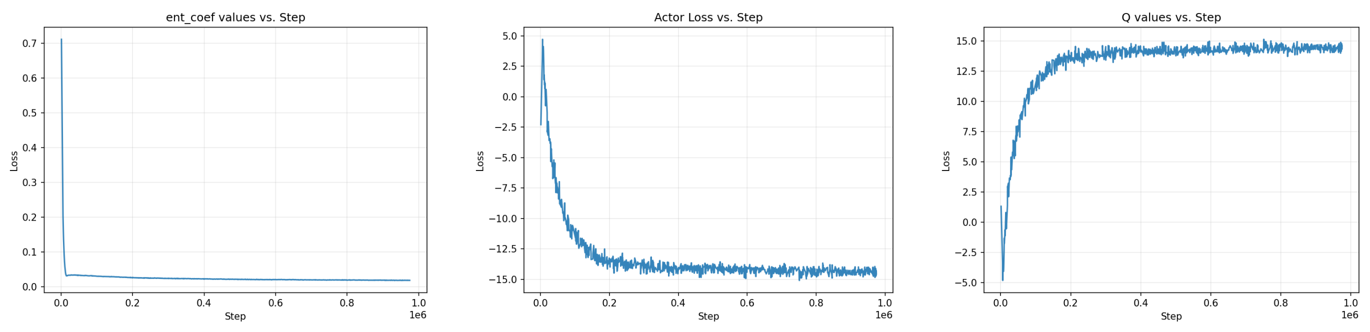
D. Discussion on Robustness

The results highlight the efficacy of distributional reinforcement learning for navigation. By explicitly truncating the upper tail of the return distribution, the TQC agent maintains conservative value estimates, avoiding the aggressive behaviors often seen in standard actor-critic methods near obstacles. Furthermore, the zone-based collision penalty provides dense, directional feedback that superiorly guides the agent in tight spaces compared to scalar distance penalties. This combination of distributional value estimation and safety-aware reward shaping yields a policy that is both efficient and robust across randomized configurations.

VI. Discussion

A. Behavioral Analysis

The learned policy demonstrates distinct behaviors driven by the shaped reward structure. In open spaces, progress



(a) Entropy α (b) Actor loss (c) Q values

Fig. 2: Training curves showing stable convergence of entropy coefficient, actor loss, and Q-values.

and heading rewards dominate, leading to rapid acceleration. Conversely, near obstacles, the zone-based proximity penalties induce proactive deceleration and smooth turning maneuvers. This graded feedback mechanism effectively prevents the "stop-and-go" oscillations often observed in sparse-reward settings, ensuring efficient yet safe navigation.

B. *Role of Distributional RL*

TQC proves essential for stability in randomized environments by explicitly modeling return variance. Unlike standard actor-critic methods prone to value overestimation in cluttered settings, our agent utilizes quantile truncation to maintain conservative value estimates. This mechanism keeps Q-values bounded and, combined with automatic entropy tuning, achieves a robust balance between exploration and exploitation.

C. *Limitations and Sim-to-Real Gap*

Despite simulation success, several limitations persist. First, the Gazebo physics approximation neglects complex wheel slippage and assumes perfect localization, underestimating real-world sensor noise and drift. Second, the current state representation relies on local history, which may be insufficient for complex mazes requiring global memory (*e.g.*, long short-term memory (LSTM) or global maps). Finally, real-world deployment would require additional safety layers, such as control barrier functions (CBF) [15], to bridge the gap between simulated rewards and physical safety constraints.

VII. Conclusion and Future Work

This paper presented an end-to-end framework for the autonomous navigation of a skid-steer mobile robot using truncated quantile critics (TQC) [6]. By integrating a realistic ROS 2–Gazebo simulation stack with a simulated 3D Ouster LiDAR, we formulated the navigation task as a continuous control problem. We proposed a comprehensive reward function incorporating a zone-based collision detection scheme, which was shown to effectively balance goal-directed progress with rigorous safety requirements. Empirically, the TQC-based policy demonstrated robust performance, achieving high success rates in randomized environments with dense obstacles. The distributional critic design, coupled with quantile truncation, successfully mitigated the value overestimation bias, leading to stable convergence and smooth trajectories. The modular implementation, separating the environment, training, and testing nodes within the ROS 2 ecosystem, provides a reusable baseline for future research in robotic reinforcement learning.

Future work will focus on three key directions:

1) **Dynamic Environments:** Extending the simulation to include moving obstacles and human actors to evaluate the policy's adaptability to time-varying hazards.
2) **Ablation Studies:** Conducting systematic comparisons against baseline algorithms (*e.g.*, SAC [10], TD3 [11]) and analyzing the impact of specific components like prioritized replay and zone-based rewards.
3) **Sim-to-Real Transfer:** Deploying the trained policy on a physical Bunker robot. This will involve domain randomization techniques to handle sensor noise and the implementation of a safety-filter layer for real-world validation.

Acknowledgment

This study was supported by the BK21 Four project funded by the Ministry of Education, Korea (4199990113966)

References

[1] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, 1997.

[2] C. Rösmann, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *Proc. 7th German Conf. Robotics*, 2012.

[3] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017, pp. 31–36.

[4] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Sci. Robot.*, vol. 7, no. 66, p. eabm6239, 2022.

[5] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2004, pp. 2149–2154.

[6] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov, "Controlling overestimation bias with truncated mixture of continuous distributional quantile critics," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 5556–5566.

[7] J. Tobin *et al.*, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017, pp. 23–30.
[8] V. Mnih, K. Kavukcuoglu, and D. Silver, “Playing Atari with deep reinforcement learning,” 2013, arXiv:1312.5602. [Online]. Available: https://arxiv.org/abs/1312.5602
[9] T. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.
[10] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1861–1870.
[11] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1587–1596.
[12] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, “Implicit quantile networks for distributional reinforcement learning,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1104–1113.
[13] G. Barth-Maron *et al.*, “Distributional policy gradients,” in *Int. Conf. Learn. Represent. (ICLR)*, 2018.
[14] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.
[15] A. D. Ames *et al.*, “Control barrier functions: Theory and applications,” in *Proc. 18th Eur. Control Conf. (ECC)*, 2019, pp. 3539–3546.