n-Step Upgrade to Quantile Regression Deep Q-Network Yields Consistent Gains

Vidusha De Silva
Department of Computer Science and Engineering
University of Moratuwa
Colombo, Sri Lanka
vidusha.21@cse.mrt.ac.lk

Uthayasanker Thayasivam
Department of Computer Science and Engineering
University of Moratuwa
Colombo, Sri Lanka
rtuthaya@cse.mrt.ac.lk

***Abstract*—Quantile Regression DQN (QR-DQN) is a strong baseline for value-based reinforcement learning, but most open-source implementations on classic control tasks still use a 1-step temporal-difference (TD) target. We revisit QR-DQN on `CartPole-v1` and evaluate a minimal change: replacing the 1-step target with an n-step return (we use n=3) while keeping the network, optimizer, exploration schedule, and evaluation protocol unchanged. This modification increases the effective propagation horizon of learning updates without altering the distributional loss (pinball) or action selection. Under a fixed training budget of 50,000 environment steps and greedy evaluation (ε=0), the n-step QR-DQN shows consistently faster learning and higher final returns across multiple random seeds compared to the 1-step baseline. We provide a small, self-contained implementation and report training curves and seed-wise statistics to facilitate reproducibility. The results suggest that adopting n-step targets is a low effort, high impact improvement for distributional value methods even on small classic-control benchmarks like `CartPole`.**

***Index Terms*—Deep reinforcement learning, distributional RL, Quantile Regression DQN (QR-DQN), n-step returns, bootstrapped targets, off-policy learning, experience replay, sample efficiency, greedy evaluation, classic control, `CartPole-v1`.**

I Introduction

Deep reinforcement learning (DRL) has achieved strong performance across control and game-playing benchmarks by combining value-based learning with scalable function approximation and replay [1], [2]. Among value-based methods, *distributional RL* estimates a full return distribution rather than only its expectation, leading to improved stability and data efficiency in practice [3], [4]. Quantile Regression DQN (QR-DQN) is a particularly practical instantiation that approximates the return distribution with a set of quantile atoms trained via the pinball (quantile) loss [4].

Despite these advances, standard QR-DQN implementations commonly employ one-step temporal-difference (TD) targets, which can slow propagation of reward information and exacerbate bootstrapping bias in sparse or delayed-reward settings [2]. Multi-step TD targets are a classical remedy that trades reduced bias for potentially higher variance and have proven beneficial for many value-based agents [1], [7]. However, systematic, minimalist evaluations of n-step targets *within* QR-DQN—holding all other components constant—remain comparatively underreported relative to larger "bundled" agents (e.g., Rainbow) that modify multiple design axes at once [7].

This paper revisits QR-DQN with a single, surgical change: replace the one-step target with an n-step return (we study n=3), while preserving the QR-DQN architecture, optimizer, exploration schedule, target network updates, and uniform replay [1], [4], [8]. On the classic-control `CartPole-v1` benchmark, we conduct a controlled multi-seed study and observe consistent gains in greedy evaluation: across four seeds, the modified agent improves mean return by $\approx$**+215** (mean Δ), with median improvement $\approx$**+219** and reduced variance, under identical training budgets and evaluation protocols.

Our study is intentionally narrow in scope. Rather than introduce additional bells and whistles such as prioritized replay [9], double estimators [8], dueling networks [10], or noisy exploration [11], we isolate the effect of n-step bootstrapping inside the quantile-regression framework. This isolation clarifies how faster credit propagation complements distributional targets, offering a simple drop-in change that is straightforward to reproduce.

Contributions. (i) We present a minimal QR-DQN variant that swaps one-step TD for n-step returns while keeping all other components fixed [4]. (ii) We provide a careful multi-seed `CartPole-v1` evaluation showing substantial and consistent performance gains and improved sample efficiency under greedy testing [12]. (iii) We release concise, self-contained code and evaluation utilities to facilitate reproduction and further ablations (e.g., sensitivity to n and target-update cadence).

II Related Work

Distributional value learning. Modeling the *distribution* of discounted returns instead of only its expectation has led to stronger value-based agents [3]. QR-DQN [4] replaces categorical supports with quantile regression, enabling optimization in (smoothed) Wasserstein metrics via the pinball / quantile-Huber loss and removing C51's projection step. IQN further learns a continuous quantile function by sampling quantile fractions at train time [5]. Going beyond fixed quantile fractions, Fully Parameterized Quantile Functions (FQF) learn both the fractions and the values end-to-end, adaptively allocating capacity over the support [14]. Our work stays strictly within the QR-DQN family, changing only the *target definition* (1-step vs. n-step) while holding the distributional head and the greedy policy unchanged.

Multi-step bootstrapping in deep RL. Multi-step TD targets speed credit propagation at the cost of higher variance [2]. In deep value-based control, n-step targets are a standard component and were a key factor in Rainbow's gains [7]. A rich literature studies the design space that n-step opens: TD(λ), Expected SARSA, Tree-Backup, and Q-learning can be unified by a single parameter σ in Q(σ) [17], clarifying bias–variance trade-offs as one interpolates between expectation and sampling. Off-policy use of multi-step targets generally requires corrections; Retrace(λ) offers a safe, truncated-IS alternative well-suited to experience replay [6], and prior work has also examined reward-level corrections for TD(λ) in off-policy settings [18]. For the low-stochasticity, ε-greedy setting of `CartPole-v1`, we find small n (here n=3) gives consistent gains without additional correction machinery. Our ablation isolates n-step inside QR-DQN rather than bundling it with architectural or exploration changes.

Historical context on n-step and λ-returns. Multi-step TD has deep roots in classical RL: TD(λ) unifies n-step targets via exponentially decayed eligibility traces [21]. For control, Watkins's Q-learning [22] and Peng's Q(λ) [23] explored trace-based credit assignment with different off-policy treatments (e.g., trace cutting vs. replacing). Tree-Backup and related backups formalize expectation-style targets without importance sampling [19]. Modern deep agents revisit these design choices in replay-based settings where short horizons (e.g., n=3–5) often strike a favorable bias–variance trade-off.

Bundled agents vs. isolated ablations. Rainbow combined multiple axes—double Q, dueling heads, prioritized replay, noisy nets, distributional value learning, and n-step—making it challenging to attribute gains to any one factor [7]. Subsequent reproducibility guidance recommends controlled, multi-seed comparisons with one change at a time [13]. We follow this guidance by swapping only the temporal-difference horizon while keeping network, optimizer, target-update cadence, replay scheme, and exploration schedule fixed.

Risk-sensitive objectives and quantiles. Distributional methods naturally expose tail behavior that risk-aware criteria (e.g., CVaR) target [3]. CVaR-based control motivates learning lower quantiles accurately [20], and quantile parametrizations such as QR-DQN/FQF are a convenient substrate when one wants to read off tail summaries or plug risk measures into action selection. Although we evaluate with risk-neutral greedy policies (mean over quantiles), our n-step modification is orthogonal and directly applicable to risk-sensitive evaluation.

Positioning. Closest to our work is Rainbow's use of n-step inside a distributional head [7]; in contrast, we ablate n-step *alone* within QR-DQN and quantify its effect on a classic-control benchmark. Our findings support the practical advice that small n (e.g., 3–5) is a low-effort, high-impact upgrade for distributional value agents before reaching for heavier additions (PER, dueling, or noisy nets).

III Method

III-A Problem Setup

We consider standard episodic reinforcement learning with discrete actions. At time t, the agent observes $s_t \in \mathcal{S}$, selects $a_t \in \mathcal{A}$, receives reward $r_t \in \mathbb{R}$, and transitions according to dynamics $p(s_{t+1} \mid s_t, a_t)$. The objective is to maximize the expected discounted return $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ with discount $\gamma \in (0, 1)$ [2]. We instantiate and evaluate the method on `CartPole-v1` from Gym [12].

TABLE I
POSITIONING OUR ABLATION AMONG COMMON VALUE-BASED BASELINES.

Agent	**Dist.**	**n-step**	**Double**	**Dueling**	**PER**	**Noisy**	**Notes**
QR-DQN (1-step)	✓	–	✓	–	–	–	Baseline in this paper
QR-DQN (n=3)	✓	✓	✓	–	–	–	Our single-change variant
IQN	✓	optional	✓	–	optional	optional	Continuous quantiles [5]
Rainbow	✓	✓	✓	✓	✓	✓	Bundled agent [7]
FQF	✓	optional	✓	–	optional	optional	Learned fractions [14]

III-B Baseline: Quantile Regression DQN (QR-DQN)

Our baseline follows the distributional RL formulation of QR-DQN [4]. Instead of learning a scalar action-value, the agent learns the return *distribution* via N quantiles per action. Concretely, the network f_θ maps s to $\mathbf{Z}_\theta(s,a) \in \mathbb{R}^N$ for each $a \in \mathcal{A}$, where $\mathbf{Z}_\theta(s,a) = [Z_\theta^{(1)}, \ldots, Z_\theta^{(N)}]$. Action selection is greedy w.r.t. the mean of quantiles:

$$a^\star(s) = \arg\max_{a\in\mathcal{A}} \frac{1}{N}\sum_{i=1}^{N} Z_\theta^{(i)}(s,a). \tag{1}$$

Network.

For `CartPole` (vector observations), we use an MLP with two hidden layers of width 128 and ReLU activations, outputting $|\mathcal{A}| \times N$ quantiles. We set $N = 51$ midpoint quantile levels $\tau_i = (i - 0.5)/N$, $i = 1, \ldots, N$ [4].

Target and Double Q-learning.

We adopt the Double Q-learning bootstrap [8]: the next action $a_{t+1}^{\text{sel}} = \arg\max_a \frac{1}{N}\sum_i Z_\theta^{(i)}(s_{t+1}, a)$ is selected by the *online* network, while target quantiles are gathered from the *target* network $f_{\bar\theta}$ at this action. A hard target update copies online parameters every τ=1000 environment steps, as in DQN [1].

Loss.

Given predicted quantiles $\hat{\mathbf{Z}} = \mathbf{Z}_\theta(s_t, a_t) \in \mathbb{R}^N$ and target quantiles $\mathbf{T} \in \mathbb{R}^N$, we minimize the *quantile regression* (pinball) loss [4]:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} \rho_{\tau_i}\big(T^{(j)} - \hat{Z}^{(i)}\big), \quad \rho_\tau(u) = \big|\tau - \mathbf{1}\{u<0\}\big|\,|u|. \tag{2}$$

We use the pure pinball (L1) variant (κ=0).

III-C Proposed Modification: n-step Bootstrapping

The sole algorithmic change from the baseline is to replace the 1-step TD target with an n-step return prior to bootstrapping:

$$R_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k}, \tag{3}$$

$$\mathbf{T} = R_t^{(n)}\mathbf{1} + \gamma^n\Big(1 - d_t^{(n)}\Big)\,\mathbf{Z}_{\bar\theta}\big(s_{t+n}, a_{t+n}^{\text{sel}}\big). \tag{4}$$

where $d_t^{(n)} \in \{0,1\}$ flags whether a terminal was encountered within the n-step window. This modification is motivated by prior evidence that n-step targets can accelerate learning and stabilize value estimation when combined with replay and target networks [2], [7]. In our experiments we keep the entire pipeline fixed and vary only $n \in \{1,3\}$, showing consistent gains with n=3.

III-D Replay, Exploration, and Optimization

We use uniform experience replay [1] with capacity 100,000, mini-batches of size 64, and learning starts after 1000 steps. Exploration follows ε-greedy with a linear schedule from 1.0 to 0.01 over 25,000 steps, then held constant, as in [1]. We optimize with Adam (lr 5×10^{-4}, $\epsilon = 10^{-8}$) and clip global gradient norm at 10.

III-E Training and Evaluation Protocol

Each run trains for 50,000 environment steps with $\gamma = 0.99$ and hard target updates every τ=1000 steps. Progress diagnostics compute quick greedy rollouts at episode boundaries. Final evaluation reports greedy (ε=0) returns averaged over multiple episodes with fixed seeds.

IV Experimental Setup

IV-A Environment and Task

All experiments are conducted on `CartPole-v1` from Gym/Gymnasium [12]. Observations are 4-D vectors;

actions are binary. Episodes terminate upon threshold violations or at horizon. Discount $\gamma = 0.99$.

IV-B Algorithms Compared

QR-DQN (1-step): QR-DQN [4] with Double Q-learning [8], uniform replay [1], hard target updates.
QR-DQN (**n=3**): identical to baseline except the bootstrap target uses n-step return with n=3 [2], [7].

IV-C Network and Optimizer

Two-layer MLP (hidden sizes 128, ReLU), output $|\mathcal{A}|\times 51$ quantiles. Adam (lr 5×10^{-4}, $\epsilon = 10^{-8}$), grad-norm clip 10.

IV-D Replay, Targets, Exploration

Uniform replay capacity 100,000, batch size 64, learn start 1000 steps. Target updates every 1000 steps. Linear ε-greedy from 1.0 to 0.01 over 25,000 steps.

IV-E Training Budget, Evaluation, Seeds

Each run: 50,000 steps. Final performance: greedy (ε=0) averages over many episodes. We summarize across seeds with mean/median/std following reproducibility guidance [13]. For the 1000-episode seed-wise A/B comparison we also compute Δ = n=3 − 1-step.

V Results

We compare the baseline 1-step QR-DQN against the proposed n-step variant with n=3 on `CartPole-v1`. Unless stated otherwise, all numbers are for *greedy* evaluation (ε=0) with fixed seeds. We report (i) large-sample evaluation over 1000 episodes per seed using saved checkpoints, and (ii) the final 10-episode greedy probes during training.

V-A Evaluation over 1000 Episodes per Seed

For each checkpoint/seed, we run 1000 greedy episodes and record the mean and episode SD. Fig. 1 visualizes per-seed performance; Fig. 2 shows $\Delta = (n{=}3) - (1\text{-step})$.

V-B Final 10-Episode Training Probes

Table III reports final greedy probes (10 episodes) during training for five seeds. The n=3 agent nearly doubles mean return and shows lower cross-seed variance.

V-C Takeaways

Using n=3 inside the QR-DQN target yields consistent per-seed improvements (Table II, Fig. 2), higher final probes with lower cross-seed variance (Table III), and faster learning without changing the network, optimizer, or exploration schedule.

TABLE II
GREEDY EVALUATION OVER 1000 EPISODES PER SEED. MEAN ± EPISODE SD; Δ IN ABSOLUTE RETURN.

Seed	**1-step**	n=3	Δ
7	274.56 ± 103.00	389.80 ± 102.69	+115.24
8	185.57 ± 14.33	387.16 ± 93.93	+201.60
9	113.23 ± 5.31	418.70 ± 93.68	+305.46
10	150.66 ± 3.97	386.92 ± 112.43	+236.25

Δ summary: mean = 214.64, median = 218.93, SD = 79.09, n=4.

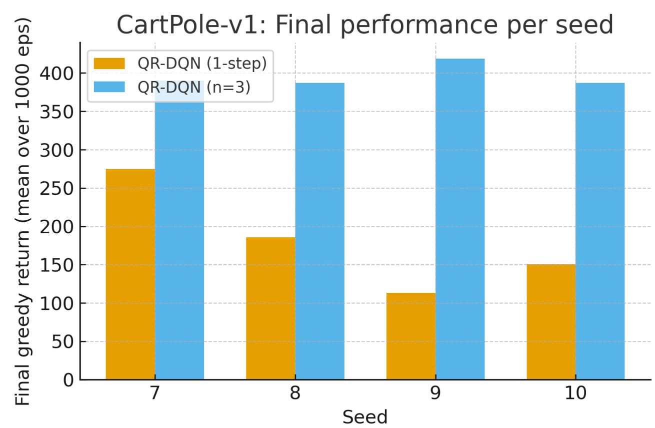

Fig. 1. Per-seed mean returns over 1000 greedy episodes on `CartPole-v1`. Bars show the mean; whiskers denote the episode standard deviation.

VI Discussion & Limitations

VI-A Why does n=3 help on `CartPole`?

Replacing 1-step TD with n-step (n=3) yields large, consistent gains for QR-DQN on `CartPole-v1`. Short multi-step returns reduce bootstrapping bias while keeping variance manageable on a low-stochasticity MDP [2]. In the distributional setting, multi-step targets propagate more informative quantile distributions, sharpening action ordering when taking the mean over quantiles [3], [4]. Gains appear early and without instability, consistent with Rainbow's observations [7].

VI-B Relationship to experience prioritization

Preliminary tests with prioritized replay (PER) suggested improvements over uniform replay, but the n=3 modification remained the dominant factor across seeds. PER's benefits can be sensitive to its hyperparameters (priority exponent, importance-sampling correction) [9]; a full

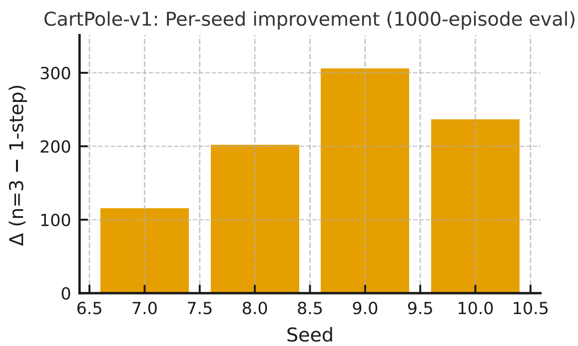

Fig. 2. Per-seed improvement Δ of QR-DQN with n=3 over the 1-step baseline on `CartPole-v1`. Bars show the mean return difference over 1000 greedy episodes; whiskers denote the episode standard deviation.

TABLE III
FINAL TRAINING PROBES (10 GREEDY EPISODES). PER-SEED MEANS AND CROSS-SEED SUMMARY (MEAN ± SD).

Seed	**1-step**	n=3
7	269.8	390.8
8	182.6	353.2
9	172.7	356.2
10	114.1	391.3
11	151.4	343.2
Summary	178.12 ± 57.59	366.94 ± 22.53

sweep is left to future work.

VI-C Threats to validity and future work

Scope is limited to `CartPole-v1`; results may differ on sparse-reward or image-based tasks. We did not extensively tune learning rate, target update period, or discount per n. Future work: ablate $n \in \{1, 2, 3, 5\}$, γ, target-update cadence, and quantile count; combine with PER/noisy nets; extend to MinAtar/Atari; and report significance tests and wall-clock efficiency [13].

VII Conclusion

This work shows that a *single*, low-friction change replacing the 1-step TD target with an n-step return produces *consistent and sizable* gains for QR-DQN on `CartPole-v1`. Holding the network, optimizer, exploration, and replay constant, the n=3 variant learns faster, attains substantially higher greedy performance at a fixed budget of 50,000 steps, and exhibits lower cross-seed variance. These results reinforce long-standing intuition about multi-step credit propagation while clarifying its *isolated* value inside a distributional RL agent.

Practically, our findings suggest a simple guideline for practitioners already using QR-DQN on classic-control or similar low-stochasticity tasks: start with a small n (e.g., n=3) before considering more invasive additions (PER, dueling heads, or noisy nets). The modification is easy to implement, computationally negligible, and robust across seeds in our study. Future work should probe sensitivity to n and γ, interactions with target-update cadence and replay schemes, and generalization beyond `CartPole-v1` to MinAtar/Atari domains.

Code and Data Availability

Repository: https://github.com/aaivu/In21-S7-CS4681-AML-Research-Projects/tree/main/projects/210099V-Offline-RL

References

[1] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.
[3] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proc. ICML*, 2017, pp. 449–458.
[4] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proc. AAAI*, 2018, pp. 2892–2901.
[5] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," in *Proc. ICML*, 2018, pp. 1096–1105.
[6] R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare, "Safe and efficient off-policy reinforcement learning," in *Proc. NeurIPS*, 2016, pp. 1054–1062.
[7] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. AAAI*, 2018, pp. 3215–3222.
[8] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI*, 2016, pp. 2094–2100.
[9] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. ICLR*, 2016.
[10] Z. Wang *et al.*, "Dueling network architectures for deep reinforcement learning," in *Proc. ICML*, 2016, pp. 1995–2003.
[11] M. Fortunato *et al.*, "Noisy networks for exploration," in *Proc. ICLR*, 2018.
[12] G. Brockman *et al.*, "OpenAI Gym," *arXiv:1606.01540*, 2016.
[13] P. Henderson *et al.*, "Deep reinforcement learning that matters," in *Proc. AAAI*, 2018, pp. 3207–3214.
[14] Z. Yang, Y. Zhang, K. Xu, and J. Wang, "Fully parameterized quantile function for distributional reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 13376–13387.
[15] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. ICML*, 2016, pp. 1928–1937.

[16] L. Espeholt *et al.*, "IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures," in *Proc. ICML*, 2018, pp. 1406–1415.
[17] K. De Asis, F. Hernandez-Garcia, G. Holland, and R. S. Sutton, "Multi-step reinforcement learning: A unifying algorithm," in *Proc. AAAI*, 2018, pp. 2830–2836.
[18] A. Harutyunyan, M. G. Bellemare, T. Stepleton, and R. Munos, "Q(λ) with off-policy corrections," in *Proc. NeurIPS*, 2016, pp. 1754–1762.
[19] D. Precup, R. S. Sutton, and S. Singh, "Eligibility traces for off-policy policy evaluation," in *Proc. ICML*, 2000, pp. 759–766.
[20] Y. Chow, A. Tamar, S. Mannor, and M. Pavone, "Risk-sensitive and robust decision-making: a CVaR optimization approach," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015, pp. 1522–1530.
[21] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, 3(1):9–44, 1988.
[22] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, 8(3–4):279–292, 1992.
[23] J. Peng and R. J. Williams, "Incremental multi-step Q-learning," *Machine Learning*, 22(1–3):283–290, 1996.