arXiv:2506.05716v1 [cs.LG] 6 Jun 2025

Ensemble Elastic DQN: A novel multi-step ensemble approach to address overestimation in deep value-based reinforcement learning

Adrian Ly, Richard Dazeley, Peter Vamplew, Francisco Cruz and Sunil Aryal

June 13, 2025

Abstract

While many algorithmic extensions to Deep Q-Networks (DQN) have been proposed, there remains limited understanding of how different improvements interact. In particular, multi-step and ensemble style extensions have shown promise in reducing overestimation bias, thereby improving sample efficiency and algorithmic stability. In this paper, we introduce a novel algorithm called Ensemble Elastic Step DQN (EEDQN), which unifies ensembles with elastic step updates to stabilise algorithmic performance. EEDQN is designed to address two major challenges in deep reinforcement learning: overestimation bias and sample efficiency. We evaluated EEDQN against standard and ensemble DQN variants across the MinAtar benchmark, a set of environments that emphasise behavioral learning while reducing representational complexity. Our results show that EEDQN achieves consistently robust performance across all tested environments, outperforming baseline DQN methods and matching or exceeding state-of-the-art ensemble DQNs in final returns on most of the MinAtar environments. These findings highlight the potential of systematically combining algorithmic improvements and provide evidence that ensemble and multi-step methods, when carefully integrated, can yield substantial gains.

1 Introduction

Overestimation bias has been a pervasive phenomenon in value-based deep reinforcement learning algorithms. The phenomenon emerges when the agent generates inflated Q-values; these may misdirect the agent to select inefficient or sub-optimal actions, leading to convergence to poor policies – this issue was first identified in Deep Q-learning (DQN) [MKS+15]. However, overestimation bias is not unique to DQN and has also been observed across other algorithms that use value-based techniques for learning [FHM18, NBM+24]. The phenomenon is well documented and has been largely attributed to the usage of the *max* operator in the value-based update rule – when an agent makes Q-values value predictions, the *max* operator is used to select actions with the highest Q-values associated with it to update the network, which may lead the agent to bias its approximations towards higher values. When this bias is coupled with the recursive nature of the algorithm, small positive errors can be deeply rooted in the agent, leading to suboptimal policy convergence [MKS+15, TS14, HMVH+18, JHS21, HH20, VHDS+18]. Inflated Q-values may also contribute to poor policy exploration as agents are more likely to explore state-action regions that have been affected by overestimation bias ([ZBX+25]). To address this issue in DQN, a number of promising heuristic approaches have been developed to reduce the effect of overestimation in order to improve the performance of the algorithm. In particular, the use of ensemble-based methods and multi-step methods has shown strong efficacy across different reinforcement learning problems.

Ensemble-based algorithms address the problem of overestimation in DQN through the use of multiple independent approximators to modulate Q-value estimations. Algorithms like Average DQN [ABS17] and MaxMinDQN [LPFW20] aggregate the predictions of multiple parallel Q-value approximators; the combination of multiple Q-networks reduces the effect of overestimated values from any one Q-network's outliers, thereby smoothing out inflated estimates and providing more reliable targets for learning.

On the other hand, multi-step methods such as Rainbow [HMVH+18] or Elastic Step DQN [LDV+24] bootstrap returns over a longer horizon and reduce the reliance on single-step bootstrapping, which

has shown reductive effects on overestimation in DQN and strong sample efficiency by accelerating credit assignment [HGS19, YYG+19, HMVH+18, LDV+24].

In general, research around the integration of different algorithmic extensions to improve performance is scarce. While heuristic approaches like ensemble-based methods and multi-step algorithms show a lot of promise, they have largely been explored in isolation. Both areas of research improve algorithmic performance through the modulation of state-action value, albeit through different methods. This raises the question of whether these approaches are complementary in the context of alleviating Q-value overestimation. While the benefits are clear, the combination of the two heuristic approaches is non-trivial, as the reductive effects of the combination of both approaches can lead to underestimation bias.

Elastic Step DQN introduced by [LDV+23, LDV+24] is an adaptive extension of multi-step DQN that dynamically adjusts the bootstrap horizon based on the similarity between consecutive states, allowing it to incorporate more distant future rewards in stable regions while shortening the horizon in rapidly changing ones. The adaptiveness of the algorithm allows it to include both single and multi-step returns; however, the single step returns are still susceptible to overestimation bias. This paper introduces Elastic Ensemble DQN (EEDQN), a novel deep reinforcement learning algorithm that builds on the concepts introduced in Elastic Step DQN and a dynamic ensemble-based aggregation method to address the overestimation bias of DQN and to improve sample efficiency. The contribution of this paper is three-fold:

- Introduces EEDQN, a novel algorithm that integrates elastic multi-step returns and ensemble learning to address overestimation bias and improve sample efficiency.
- Conducts a comprehensive ablation study to evaluate the individual contributions of the multi-step learning and ensemble components of EEDQN.
- Benchmarks the algorithmic performance of EEDQN against established algorithms such as Double DQN (DDQN), Averaged DQN (AvgDQN), Maxmin DQN, Elastic Step DQN (ESDQN), n-step DQN, and DQN to demonstrate its effectiveness in the MinAtar environment.

2 Related works

While both ensemble-based learning and multi-step learning have shown promise in the alleviation of overestimation bias in value-based deep reinforcement learning, there has been little research in the integration of ensemble and multi-step learning. To the best of our knowledge at the time of writing, there has only been one other paper that combines multi-step learning and ensemble base learning ([CH22]). However, [CH22] combines ensemble and multi-step learning in the context of hierarchical reinforcement learning, using an actor-critic with a dual training phase architecture (high-level and low-level phases). At the low-level phase, base learners are trained by multi-step where each base learner also adjusts its policy parameters by incorporating information from randomly selected peer learners based off of a bootstrap value. In the high-level training phase, a global critic function is used to evaluate the performance of the base learners. Fundamentally, the approach is significantly different from EEDQN which combines ensemble learning in a parallel manner, with the ensemble aggregation method based on the step size.

One of the earlier attempts to address overestimation was through the algorithm Double DQN ([VHGS16]) which sought to decouple the action-selection step of the algorithm from the action evaluation step, which led to more stable training in DQN. Other methods like Average DQN ([ABS17]) which preceded Double DQN ([VHGS16])showed that it was able to reduce variance through the averaging of multiple Q-network predictions, which effectively smoothed out major fluctuations caused by stochasticity during the training of the agent. While the averaging function has shown to improve overestimation bias, it does not eliminate the prospect of overestimation – Maxmin DQN ([LPFW20]) takes a much more conservative approach by taking the minimum Q-value from an ensemble.

Standard multi-step algorithms like n-step DQN [LDV+24, OCC20] bootstrap return over a predetermined and constant horizon which has been shown empirically to lead to suboptimal performance in environments where the temporal dynamics of the environment may vary [LDV+24].

3 Ensemble Elastic DQN

3.1 Elastic Step DQN

Traditional multi-step DQN methods fix the number (n) of steps to bootstrap, which was a difficult hyper-parameter to tune and was shown empirically to not learn as well as Elastic Step DQN ([LDV+24]) which was able to vary the number of steps to bootstrap. The implementation in the original paper used the hidden layers in the Q-networks to extract a representation of the states at every timestep. The algorithm would then use a clustering algorithm (HDSCAN was used in the original) to determine whether one state transition belonged to the same cluster. If the clustering algorithm labeled the transition as belonging in the same cluster, then it would accumulate the reward and continue the loop. On the other hand, if the clustering algorithm determined that the state transitions did not belong in the same cluster, then it would store the transition. The algorithm was shown to be an effective algorithm in the classical gym control environments; however, it struggled to scale beyond that, as clustering algorithms were very computationally expensive to run. Hence, to combine Elastic Step methods and ensemble methods, we must first solve for the scalability problem.

3.2 Scalable Elastic Step DQN

The primary purpose behind Ensemble Elastic DQN (EEDQN) is to draw on the benefits of ensemble learning combined with the regularization effects of elastic multi-step learning to improve sample efficiency and over-estimation bias. However, the combination of ensembling and elastic steps is not a straightforward one. One limitation of the original implementation of Elastic Step DQN was the scalability of the algorithm on medium to larger sized environments like the MinAtar or Atari learning environment. To solve for this, instead of using a clustering algorithm to discern similarity between states, the algorithm uses a simple formula to generate a threshold to discern similarity instead. The algorithm leverages a historical buffer B which stores the absolute differences between the state-action values of the current state (s_t) against the last updated state (S_{t+d+1}) where d represents the number of steps that have been bootstrapped. At each timestep, a dynamic threshold is calculated using the expression $h = Avg(B) + \frac{Std.dev(B)}{\sqrt{n}}$, where B denotes the buffer of recorded differences and n represents the size of the buffer B. This lightweight, adaptive threshold serves as an efficient alternative to clustering approaches, which are much more computationally demanding, thereby improving scalability in large and complex environments.

Algorithm 1 Target y_{t+d+1} Calculation

function ComputeTarget(d, s_{t+d+1}, a_t, R_t)
  if $d > 0$ **and** non-terminal **then**
    $y_{t+d+1} \leftarrow R_t + \gamma^d \min_i \hat{Q}_i(s_{t+d+1}, a_t)$
  else if $d = 0$ **and** non-terminal **then**
    $y_{t+d+1} \leftarrow R_t + \gamma^d \operatorname{Avg}_i \hat{Q}_i(s_{t+d+1}, a_t)$
  else
    $y_{t+d+1} \leftarrow R_t$
  return y_{t+d+1}

Algorithm 2 Compute h from Q-values

function ComputeThreshold(s_t, s_{t+d+1}, B, $\hat{Q}$)
  $z \leftarrow \left|\operatorname{Avg}_i \hat{Q}_i(s_t) - \operatorname{Avg}_i \hat{Q}_i(s_{t+d+1})\right|$
  $B \leftarrow B \cup \{z\}$ ▷ append to buffer
  $n \leftarrow |B$ — ▷ buffer size
  $h \leftarrow \operatorname{Avg}(B) + \dfrac{\operatorname{StdDev}(B)}{\sqrt{n}}$
  return h

Algorithm 3 Greedy action selection

function STEP(s_t, ϵ, $\hat{Q}$, *env*)
    $a_t \leftarrow \text{EPSILONGREEDY}(s_t, \hat{Q}, \epsilon)$
    $(s_{t+d+1}, r_t) \leftarrow env.\text{STEP}(a_t)$
    return (a_t, s_{t+d+1}, r_t)

Algorithm 4 Ensemble Elastic Deep Q-Network (EEDQN)

procedure EEDQN(M, T, ϵ, I, H, γ, *env*)
    Initialise replay buffer $D \leftarrow \text{REPLAYBUFFER}(I)$
    Initialise state-diff buffer $B \leftarrow \text{STATEBUFFER}(H)$
    Initialise N Q-networks $\{Q_i\}_{i=1}^{N}$ with weights θ
    Initialise target nets $\{\hat{Q}_i\}_{i=1}^{N}$ with $\theta^- \leftarrow \theta$
    Pre-fill D and B with random experience
    for episode $= 1$ **to** M **do**
        reset env and observe s_t
        $d \leftarrow 0,\ R_t \leftarrow 0$
        for $t = 1$ **to** T **do**
            $(a_t, s_{t+d+1}, r_t) \leftarrow \text{STEP}(s_t, \epsilon, \hat{Q}, env)$
            $h \leftarrow \text{COMPUTETHRESHOLD}(s_t, s_{t+d+1}, B, \hat{Q})$
            $z \leftarrow \left|\text{Avg}_i\, \hat{Q}_i(s_t) - \text{Avg}_i\, \hat{Q}_i(s_{t+d+1})\right|$
            if $z > h$ **then**
                store $(s_t, a_t, R_t, s_{t+d+1}, \gamma^d)$ in D
                $d \leftarrow 0,\ R_t \leftarrow 0$
            else ▷ still accumulating
                $d \leftarrow d + 1$
                $R_t \leftarrow R_t + \gamma^d r_t$
            if ready to update **then**
                sample mini-batch $\mathcal{B}$ from D
                for all (s, a, R, s', d) in mini batch **do**
                    $y \leftarrow \text{COMPUTETARGET}(d, s', a, R)$
                    update each Q_i by gradient descent on $(y, Q_i(s, a))$
                if target update interval is met **then**
                    $\theta^- \leftarrow \theta$ ▷ sync targets
            $s_t \leftarrow s_{t+d+1}$ ▷ move to next state

Table 1: The hyper-parameters used across the MinAtar environments

	DQN	**Avg DQN**	**Maxmin DQN**	**EEDQN**	**ESDQN**	**Double DQN**
Ensemble Count		2	2	2		
Gamma	0.99	0.99	0.99	0.99	0.99	0.99
Update Horizon	1	1	1	1	1	1
Target Update Interval	1000	1000	1000	1000	1000	1000
Learning Rate	0.00025	0.00025	0.00025	0.00025	0,.00025	0.00025
Epsilon	0.0003125	0.00031	0.0003125	0.0003	0.0003125	0.0003125
Replay Memory Size	100000	100000	100000	100000	100000	100000
Batch Size	32	32	32	32	32	32
State Memory Size				10000	10000	

3.3 EEDQN training procedure

EEDQN (algorithm 4) builds upon this foundation by maintaining multiple parallel Q-networks which have shown to mitigate overestimation bias through aggregation techniques such as averaging or minimum selection across network outputs. By integrating the adaptive multi-step update mechanism with an ensemble of learners, it allows the agent to dynamically modulate learning signals while preserving computational tractability, offering a more sample-efficient alternative to traditional ensemble or adaptive step-size approaches.

The training loop for EEDQN follows the standard ESDQN procedure with a key difference in how the Q-value predictions are aggregated from the networks. The agent begins by initializing N number of Q networks and N target networks $\hat{Q}$, replay buffer D and a state difference buffer B. The replay memory is pre-filled with random experiences while the state difference buffer is pre-filled with absolute state differences. At each timestep, the agent would observe its current state s_t, select an action using $\epsilon - greedy$, execute in the environment and observe the next state s_{t+1} and reward r – the absolute state difference is also calculated and stored in the state difference buffer. As with ESDQN, the algorithm assesses the similarity between the current state and the next to determine whether to extend the current transition trajectory through reward accumulation or process and store the experience in the replay memory.

During the training update phase, each network is updated independently of each other. A random batch of experiences is drawn from the replay memory D, and the target Q-value is generated for each sampled experience for each corresponding network and is then aggregated. However, EEDQN treats single-step experiences and multi-step experiences differently; for single-step experiences, the algorithm takes the average of the Q-value predictions $y_{t+d+1} = R_t + \gamma^d Avg(\hat{Q}_i(s_{t+d+1}, a_t))$, while for multi-step experiences, the minimum of the Q-value predictions is taken over the ensemble of networks $y_{t+d+1} = R_t + \gamma^d Min(\hat{Q}_i(s_{t+d+1}, a_t))$ where $Q^{(i)}$ denotes the target network of the i-th ensemble member, and the minimization is taken across all networks in the ensemble. This approach aims to reduce overestimation bias by penalizing overly optimistic value estimates.

The intuition to use the minimum aggregation on multi-step experiences while the mean aggregation is applied on single-step experiences is largely due to the desire to avoid overly conservative estimations that may hamper exploration [WZL+22, HH20].

4 Empirical evaluation

4.1 Experimental setup

In this experiment MushroomRL [DTB+21] was used as the reinforcement learning framework to ensure consistency and reproducibility. Baseline DQN, Average DQN, MinMaxDQN, Double DQN, and Elastic Step DQN (ESDQN) were tested against EEDQN. Each of these algorithms was evaluated using 10 distinct seeds to ensure robustness and mitigate variability due to stochastic factors during training. A permutation test was used to test for statistical significance between EEDQN and the benchmark algorithms. Agents were trained for a million timesteps, however, the reward was captured at an episodic level and aggregated into 100 epochs to smooth variation in the graphs.

Table 2: Average episodic reward achieved (last 100 episodes of training) by each algorithm across five MinAtar environments. Bolded values indicate the highest-performing algorithm for each environment. ESDQN has been included in this table as the baseline algorithm for comparison.

	Seaquest	SpaceInvaders	Asterix	Breakout	Freeway
ESDQN	3.23	61.02	13.78	23.75	24.61
EEDQN	10.78	101.42	**19.45**	**27.24**	**41.3**
VariantEEDQN	29.97	95.2	3.23	18.65	38.22
MinEEDQN	**34.76**	98.01	2.74	17.57	36.74
MeanEEDQN	9.02	100.07	17.3	25.19	38.79
ConvexEEDQN1	11.6	102.99	19.3	24.74	38.93
ConvexEEDQN2	12.22	**105.88**	14.85	23.81	37.47
ConvexEEDQN3	9.29	100.52	5.38	20.31	36.01

To measure whether an algorithm has overestimated in an environment, the maximum absolute Q-value achieved during training is compared against the theoretical Q-value upper bound for the environment which is given by $|Q|_{max} = \frac{r_{max}}{1-\gamma}$. Given that the maximum reward per timestep for the MinAtar environments is 1 with the exception of Seaquest which has a maximum reward per timestep of 10, the $|Q|_{max}$ given gamma equals 0.99 is 100 for SpaceInvaders, Asterix, Breakout and Freeway while $|Q|_{max}$ given gamma equals 0.99 for Seaquest is 1000.

To evaluate the proposed algorithm, an extensive experimental study was conducted across the 5 MinAtar environments – Seaquest, SpaceInvaders, Asterix, Breakout and Freeway [YT19, OCC20]. The MinAtar environment was selected because it provided a testbed that maintained the behavior challenges of the original Atari-like tasks without the computational expense of the ALE benchmarks. Additionally, the reduced computational complexity allowed the algorithm to be run across more seeds on a very limited computation budget. This choice aligns with the need for robust statistical significance and generalization of the empirical results presented. The hyper-parameters used were the same across all the algorithms and are consistent with the hyper-parameters from [OCC20], and $\epsilon - greedy$ was used across every algorithm for exploration and decayed linearly.

4.2 Results and discussion

Table 3 summarises the average final rewards (average reward achieved in the last 100 episodes) achieved by each algorithm across the five MinAtar environments. EEDQN consistently outperforms ESDQN and other baseline DQN variants, achieving the highest average final returns in four of the five environments (SpaceInvaders, Asterix, Breakout, and Freeway) and statistically significant results against most algorithms in the four environments with the exception of AvgDQN in Asterix and ESDQN in Breakout where the p-value was greater than 0.05. However, MaxMin DQN outperforms EEDQN in Seaquest; but, EEDQN is still the second best performing algorithm for Seaquest. Notably, across four of the MinAtar environments, the mean performance against ESDQN is statistically significant, which demonstrates the value of combining ensembles with elastic multi-steps. Furthermore, Figure 1 shows that in the 4 environments in which EEDQN performs the best, it converges to its optimal policy much faster or just as fast as the other algorithms.

Among the five benchmark environments, Freeway and Seaquest present the biggest exploration challenges, making them ideal testbeds for sparse-reward evaluation. In Freeway, the agent must cross a freeway with moving cars at randomised speeds, yet receives reward only upon a successful crossing, with no intermediate feedback and also immediately resets after. Seaquest compounds difficulty through requiring agents to complete multiple objectives – the first objective is when enemies are destroyed which yields +1 reward, the second objective is to rescue divers and safely resurface which gives 10 reward. The strong empirical performance of both Maxmin DQN and Ensemble Elastic Step DQN (EEDQN) in these domains suggests that adaptive multi-step updates and conservative ensemble estimates can improve learning under sparse and delayed reward environments.

Figure 2 shows the ratio between the max Q-value achieved in each epoch divided by the theoretical Q-value over time for each benchmark algorithm across the 5 MinAtar environments. When a line crosses past the horizontal dotted line, the algorithm has overestimated. It is interesting to note that

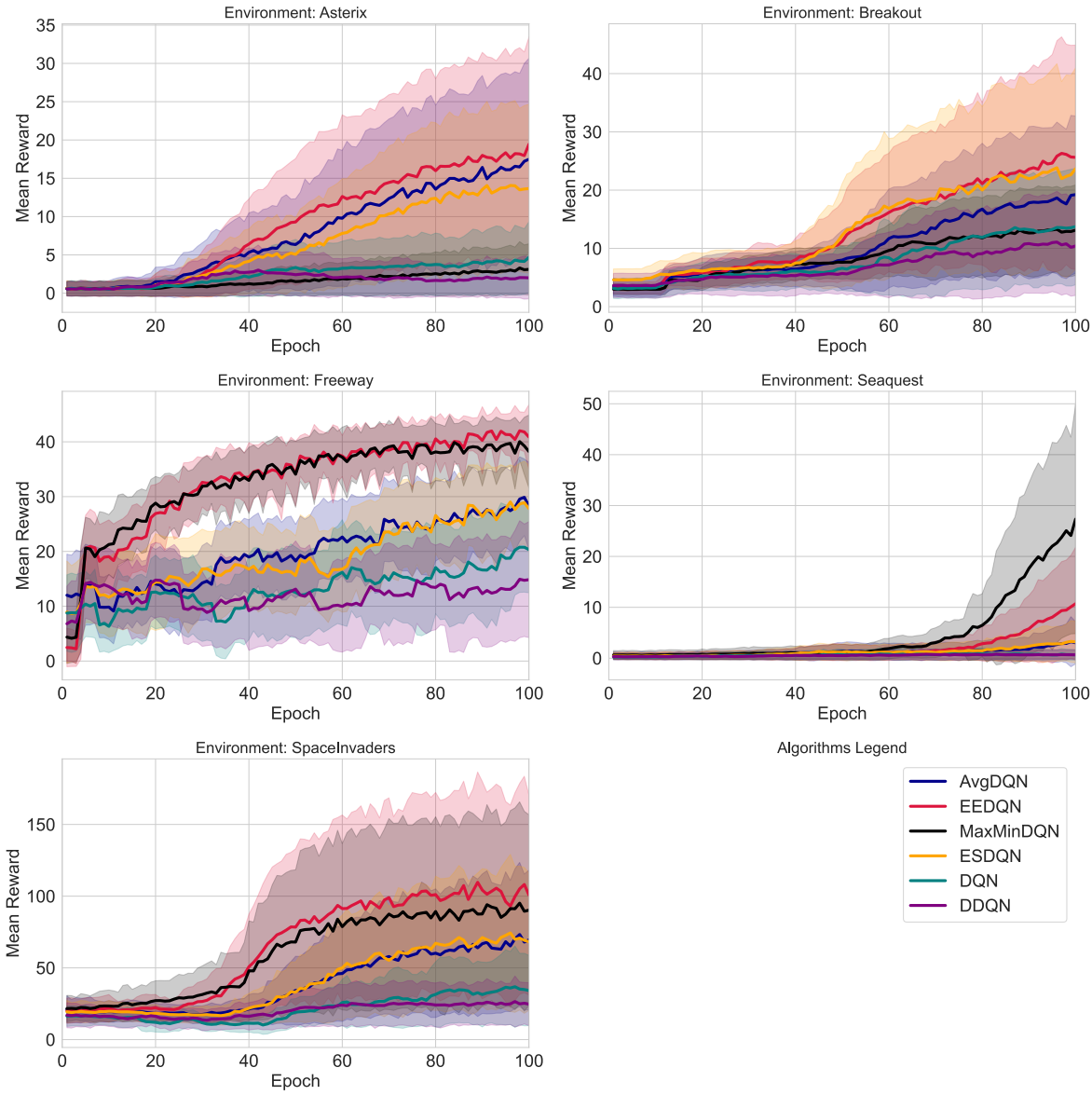


Figure 1: Training performance comparison of EEDQN and benchmark algorithms. The x-axis denotes training epochs, while the y-axis shows the mean episodic reward. Each curve represents the mean reward across seeds, and the shaded region indicates the 95 percent confidence interval around the mean.

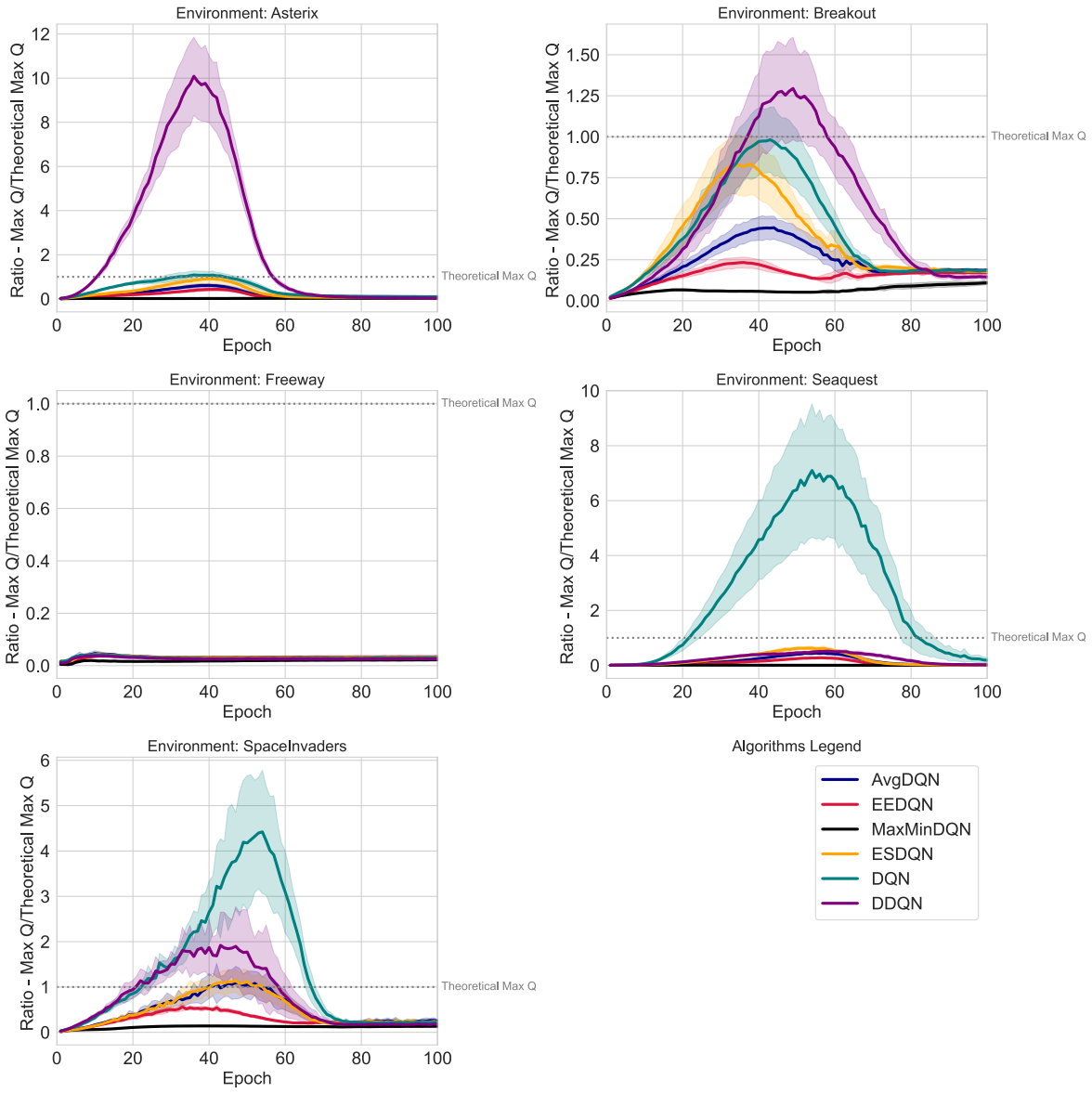


Figure 2: Absolute Q-values normalised by the maximum theoretical Q-value across training epochs for each environment. The x-axis represents training epochs, while the y-axis shows the absolute Q-value scaled such that the theoretical maximum equals to 1. Lines exceeding the value of 1 indicate overestimation. Each curve represents the mean across seeds, with shaded regions denoting the 95 percent confidence interval

EEDQN and Maxmin DQN were the only two algorithms that successfully predicted well below the theoretical max Q-value for all the environments, while ESDQN, AvgDQN, DDQN, and DQN all have overestimated at some point during learning. It is also notable that the overestimation peak almost always occurs either in earlier to mid-learning (epoch 15-50), but never during later epochs – the overestimation peak observed corresponds well with when the learning rewards begin to plateau in Figure 1 and indicates that overestimation may only occur when an agent is still learning an environment but subsides as an agent converges to a policy.

Table 3: Final performance comparison between EEDQN and benchmark algorithms, measured as the mean reward over the last 100 episodes. Bolded values indicate the best-performing algorithm for each environment. EEDQN achieves the highest final performance across all environments except Seaquest, where Maxmin DQN outperforms other methods.

	Seaquest		SpaceInvaders		Asterix		Breakout		Freeway	
	mean	p-value	mean	p-valuie	mean	p-valuie	mean	p-valuie	mean	p-valuie
AvgDQN	3.01	<0.01	68.55	<0.01	17.73	0.21	19.54	<0.01	28.6	<0.01
MaxMinDQN	**27.58**	<0.01	90.18	0.05	3.18	<0.01	13.54	<0.01	39.1	0.05
ESDQN	3.18	<0.01	68.21	<0.01	13.69	<0.01	23.86	0.26	28.26	<0.03
DQN	0.65	<0.01	33.96	<0.01	4.76	<0.01	13.81	<0.01	20.03	<0.04
DDQN	0.68	<0.01	24.71	<0.01	1.95	<0.01	10.35	<0.01	14.14	<0.05
EEDQN	10.78		**101.42**		**19.45**		**25.77**		**41.3**	

4.3 EEDQN ablation study

An ablation study was completed to assess the most effective way to combine Q-value outputs through Ensemble Elastic Step DQN (EEDQN). We conducted an ablation study examining 7 different aggregation configurations against ESDQN.

- EEDQN: For experiences that were multi-step the minimum Q-values are taken, for experiences that were single-step the average is taken
- VariantEEDQN: For experiences that were multi-step the average Q-values are taken, for experiences that were single-step the minimum is taken
- MinESDQN: The minimum is taken for all experiences
- AvgESDQN: The average is taken for all experiences
- ConvexEDQN1: 0.75 x Average Q-value for multi-step experiences + 0.25 x minimum Q-value for experiences that were single-step
- ConvexEDQN2: 0.5 x Average Q-value for multi-step experiences + 0.5 x minimum Q-value for experiences that were single-step
- ConvexEDQN3: 0.25 x average Q-value for multi-step experiences + 0.75 x minimum Q-value for experiences that were single-step

These experiments were designed to isolate the average improvement in learning performance over the elastic step mechanism. All hyper-parameters were held constant in the following experiments. The experiments evaluated each aggregation strategy across 5 MinAtar environments for 10 runs on 10 seeds. Table 2 presents the comparative performance of various Ensemble DQN variants, including the proposed Ensemble Elastic DQN (EEDQN), across five MinAtar environments – Asterix, Breakout, Freeway, Space Invaders, and Seaquest. The best-performing approach for each environment was determined by averaging the returns over the final 100 episodes. EEDQN achieved the highest average final performance in three environments, Asterix, Breakout, and Freeway. MeanEEDQN excelled in SpaceInvaders, while MinEEDQN outperformed others in Seaquest. A permutation test was performed and it indicates that EEDQN's performance improvements over the baseline ESDQN are

significant (p-value ¡ 0.05) across all environments. However, when compared to other ensemble approaches, EEDQN's performance advantages were mixed, suggesting that while EEDQN offers notable improvements over the baseline, its performance is not uniform across all ensemble configurations.

5 Conclusions and future work

In this paper, we introduced Ensemble Elastic DQN (EEDQN), a novel algorithm that combines adaptive multi-step returns with conservative ensemble estimations. Empirically, EEDQN demonstrated robust performance across the 5 MinAtar environments, and was able to successfully mitigate overestimation bias across all the MinAtar environments. Ablation studies confirmed that neither elastic steps nor ensemble alone suffice to reach the highest performance; rather, their interaction drives the largest gains in final performance, stability, and bias reduction. While our results indicate that combining adaptive multi-step updates and ensemble learning in value-based methods is effective, there remain many opportunities for future exploration. Innovations and approaches developed specifically to overcome overestimation in DQN have often been replicated in other deep reinforcement learning algorithms that also suffer from overestimation bias; hence, there are opportunities to extend this to algorithms such as actor-critic.

References

[ABS17] Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International conference on machine learning*, pages 176–185. PMLR, 2017.

[CH22] Gang Chen and Victoria Huang. Ensemble reinforcement learning in continuous spaces–a hierarchical multi-step approach for policy training. *arXiv preprint arXiv:2209.14488*, 2022.

[DTB+21] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Mushroomrl: Simplifying reinforcement learning research. *Journal of Machine Learning Research*, 22(131):1–5, 2021.

[FHM18] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[HGS19] J Fernando Hernandez-Garcia and Richard S Sutton. Understanding multi-step deep reinforcement learning: A systematic study of the dqn target. *arXiv preprint arXiv:1901.07510*, 2019.

[HH20] Qiang He and Xinwen Hou. Wd3: Taming the estimation bias in deep reinforcement learning. In *2020 IEEE 32nd international conference on tools with artificial intelligence (ICTAI)*, pages 391–398. IEEE, 2020.

[HMVH+18] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[JHS21] Jonas Jäger, Felix Helfenstein, and Fabian Scharf. Bring color to deep q-networks: Limitations and improvements of dqn leading to rainbow dqn. In *Reinforcement Learning Algorithms: Analysis and Applications*, pages 135–149. Springer, 2021.

[LDV+23] Adrian Ly, Richard Dazeley, Peter Vamplew, Francisco Cruz, and Sunil Aryal. Elastic step ddpg: Multi-step reinforcement learning for improved sample efficiency. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 01–06. IEEE, 2023.

[LDV+24] Adrian Ly, Richard Dazeley, Peter Vamplew, Francisco Cruz, and Sunil Aryal. Elastic step dqn: A novel multi-step algorithm to alleviate overestimation in deep q-networks. *Neurocomputing*, 576:127170, 2024.

[LPFW20] Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. *arXiv preprint arXiv:2002.06487*, 2020.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[NBM+24] Michal Nauman, Michał Bortkiewicz, Piotr Miłoś, Tomasz Trzciński, Mateusz Ostaszewski, and Marek Cygan. Overestimation, overfitting, and plasticity in actor-critic: the bitter lesson of reinforcement learning. *arXiv preprint arXiv:2403.00514*, 2024.

[OCC20] Johan S Obando-Ceron and Pablo Samuel Castro. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. *arXiv preprint arXiv:2011.14826*, 2020.

[TS14] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 connectionist models summer school*, pages 255–263. Psychology Press, 2014.

[VHDS+18] Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

[VHGS16] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[WZL+22] Wei Wei, Yujia Zhang, Jiye Liang, Lin Li, and Yyuze Li. Controlling underestimation bias in reinforcement learning via quasi-median operation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8621–8628, 2022.

[YT19] Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

[YYG+19] Yinlong Yuan, Zhu Liang Yu, Zhenghui Gu, Yao Yeboah, Wu Wei, Xiaoyan Deng, Jingcong Li, and Yuanqing Li. A novel multi-step q-learning method to improve data efficiency for deep reinforcement learning. *Knowledge-Based Systems*, 175:107–117, 2019.

[ZBX+25] Hongming Zhang, Fengshuo Bai, Chenjun Xiao, Chao Gao, Bo Xu, and Martin Müller. beta-dqn: Improving deep q-learning by evolving the behavior. *arXiv preprint arXiv:2501.00913*, 2025.