

ES-C51: Expected Sarsa Based C51 Distributional Reinforcement Learning Algorithm

Rijul Tandon^{a,*}, Peter Vamplew^b, Cameron Foale^b

^a*UIET, Panjab University , Chandigarh , India*

^b*Federation University, Victoria, Australia*

Abstract

In most value-based reinforcement learning (RL) algorithms, the agent estimates only the expected reward for each action and selects the action with the highest reward. In contrast, Distributional Reinforcement Learning (DRL) estimates the entire probability distribution of possible rewards, providing richer information about uncertainty and variability. C51 is a popular DRL algorithm for discrete action spaces. It uses a Q-learning approach, where the distribution is learned using a greedy Bellman update. However, this can cause problems if multiple actions at a state have similar expected reward but with different distributions, as the algorithm may not learn a stable distribution. This study presents a modified version of C51 (ES-C51) that replaces the greedy Q-learning update with an Expected Sarsa update, which uses a softmax calculation to combine information from all possible actions at a state rather than relying on a single best action. This reduces instability when actions have similar expected rewards and allows the agent to learn higher-performing policies. This approach is evaluated on classic control environments from Gym, and Atari-10 games. For a fair comparison, we modify the standard C51's exploration strategy from ϵ -greedy to softmax, which we refer to as QL-C51 (Q- Learning based C51). The results demonstrate that ES-C51 outperforms QL-C51 across many environments.

Keywords: Distributional Reinforcement Learning (DRL), C51, Expected Sarsa

*Corresponding author

Email addresses: `letscomerijul@gmail.com` (Rijul Tandon),
`p.vamplew@federation.edu.au` (Peter Vamplew), `c.foale@federation.edu.au`
(Cameron Foale)

1. Introduction

RL has demonstrated significant potential in training autonomous agents to operate effectively in stochastic environments. In standard RL, the goal is to learn an optimal policy $\pi(a|s)$ that maximizes the expected return. The action value function $Q(s, a)$ is given by equation 1

$$Q(s, a) = \mathbb{E}[G \mid s_0 = s, a_0 = a], \quad (1)$$

where $G = \sum_{t=0}^{\infty} \gamma^t R_t$ is the return (cumulative discounted rewards starting from state s and action a , following policy π), R_t and $\gamma \in (0, 1]$ denote the reward and discounting factor at time step t . However, this approach only provides the expected reward $\mathbb{E}[G]$ for each state action pair (s, a) , without revealing the variability or uncertainty in outcomes. DRL addresses this issue by modeling the return distribution in equation 2 rather than just its expectation:

$$Z^\pi(s, a) \sim P(G|s, a, \pi), \quad (2)$$

where $P(G|s, a, \pi)$ is the probability distribution over returns. This provides a complete view of the possible outcomes and their probabilities, enabling better decision making. One of the most widely used algorithms in DRL is C51, which approximates the distribution $Z^\pi(s, a)$ using a categorical distribution. C51 updates its distributions using a Q-learning based greedy policy shown by equation 3 where the next distribution is chosen from the action with the highest expected reward given by equation 4

$$Z(s, a) = R(s, a) + \gamma \times Z(s', a'). \quad (3)$$

$$a' = \arg \max_{a \in \mathcal{A}} \mathbb{E}[Z(s', a)]. \quad (4)$$

While effective in many cases, this greedy selection can fail when multiple actions have identical expected rewards but different variances. For example, consider a state s where two possible actions a_1 and a_2 satisfy equation 5

$$\mathbb{E}[Z(s, a_1)] = \mathbb{E}[Z(s, a_2)], \quad (5)$$

where $\mathbb{E}[Z(s, a)]$ is given by equation 6 where $p_\theta(s, a, z_i)$ represents the probability of atom z_i for state action pair (s, a) and N denotes the number of atoms

$$Q(s, a) = \mathbb{E}[Z(s, a)] = \sum_{i=0}^{N-1} z_i \cdot p_\theta(s, a, z_i) \quad (6)$$

yet their distributions differ in variance shown by equation 7

$$\text{Var}(Z(s, a_1)) \ll \text{Var}(Z(s, a_2)) \quad (7)$$

where variance is calculated via equation 8

$$\text{Var}(Z(s, a)) = \sum_{i=0}^{N-1} p_\theta(s, a, z_i) (z_i - \mathbb{E}[Z(s, a)])^2 \quad (8)$$

In such cases, an agent choosing a_2 might experience unstable performance due to high variance in returns. Figure 1 illustrates $Z(s, a_1)$ a stable C51 reward distribution for a given state action pair (s, a_1) .

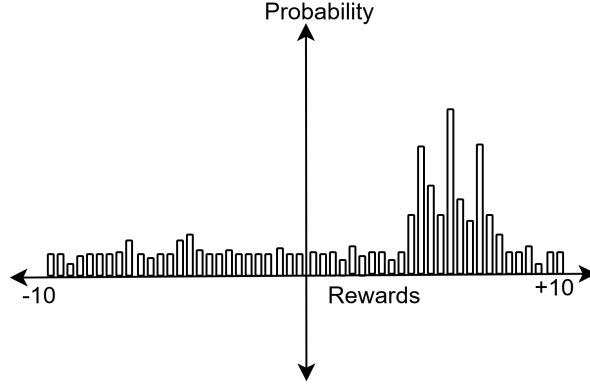


Figure 1: Example of a stable C51 distribution $Z(s, a_1)$ with low variance.

Figure 2 illustrates $Z(s, a_2)$ an unstable, high variance distribution the selection of which can lead to poor long term performance even after the learning phase.

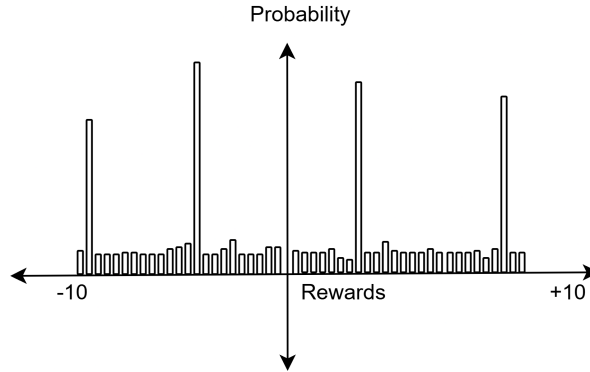


Figure 2: Example of an unstable C51 distribution $Z(s, a_2)$ with high variance.

Moreover an important source of instability arises in the value estimate for the action leading to s_t from the previous state (s_{t-1}, a) . Suppose the agent initially selects a_1 as the greedy action in s_t . Then $Z(s_{t-1}, a)$ will converge towards a discounted version of $Z(s_t, a_1)$. However, the greedy action at s_t may later switch, either due to a_1 sampling from the lower tail of its distribution or due to exploratory selection of a_2 which yields a high return. Although this change has little impact on $Q(s_t, a)$, since $Z(s_t, a_1)$ and $Z(s_t, a_2)$ share the same expectation, it can cause substantial updates to $Z(s_{t-1}, a)$. As a result, repeated switches in greedy choice may delay the stabilization of $Z(s_{t-1}, a)$. This intuition is similar to the value function interference problem identified in multi-objective reinforcement learning (MORL), where multiple values for each (s, a) pair must be combined to estimate $Q(s, a)$ and greedy action switching introduces instability [1]. It has been observed that due to ‘policy churn’ greedy action selections are subject to rapid change in deep RL, with around 10% of all states undergoing a change in greedy action after a single gradient update of DQN [2]. To address the possible instability produced in C51 by policy churn, this study proposes the ES-C51 algorithm which incorporates concepts from the Expected Sarsa algorithm [3] into C51. Instead of relying solely on the greedy choice, ES-C51 updates the current distribution using a softmax over all possible actions. The main contributions of this paper are as follows:

- **Algorithmic modification:** Study extends the standard QL-C51 algorithm by introducing ES-C51 (Expected Sarsa based C51) variant.

- **Policy and distributional updates:** Both C51 implementations were modified to employ a softmax based action selection strategy. In QL-C51, the Bellman backup remains greedy, whereas in ES-C51 the update leverages a softmax policy with a decaying temperature parameter (τ). As τ gradually approaches zero, ES-C51 naturally converges to a greedy policy.
- **Comprehensive evaluation:** Large scale experiments using 10 independent seeds across Atari-10 environments [4] (in both stochastic and deterministic variants) and two classic control Gym environments. The results were analyzed using the Wilcoxon signed rank test, and comparative performance plots were generated.
- **Performance interpretation:** This study presents a comprehensive quantitative evaluation of performance across all environments, including a detailed time efficiency comparison between both approaches. These quantify the performance gains achievable with ES-C51.

2. Literature Survey

2.1. Distributional reinforcement learning (DRL)

DRL marks a pivotal advance in the field of RL, where instead of estimating the expected return for state-action pairs, agents learn the full probability distribution of returns. This approach addresses core limitations of scalar RL, which collapses all reward information into a single expected value thereby losing crucial details about risk sensitivity and variance in returns. [5] presents a unifying framework for designing and analysing DRL algorithms in terms of recursively estimating statistics of the return distribution. Bellemare et al. [6] introduced the C51 algorithm, the first practical DRL approach, mapping returns to a categorical distribution over a fixed set of atoms using a novel projection operator. Rowland et al. [7] provided a theoretical analysis of the projected distributional Bellman operator, relating it to the Cramér distance metric and giving convergence guarantees for sample-based categorical algorithms. Kastner et al. [8] later extended this analysis to the more practical KL-divergence loss, closing the gap between theory and practice.

C51 inspired further advances, such as quantile-based approaches like QR-DQN and Implicit Quantile Networks (IQN) [9, 10] supplanted atom

based methods by directly learning quantile functions or continuous quantile parameterizations. These works highlight the value of flexible distributional parameterization, but also reveal practical challenges, such as instability and computational cost.

Recent refinements, such as Dual Expectile Quantile Regression (DEQR) [11], show that hybrid statistical measures can yield gains in stability and efficiency. Enwerem et al. [12] proposes a risk-regularized quantile-based algorithm integrating Conditional Value-at-Risk (CVaR) to enforce safety without complex architectures and provide theoretical guarantees on the contraction properties of the risk-sensitive distributional Bellman operator in Wasserstein space, ensuring convergence to a unique cost distribution. Shaik et al. [13] introduced Distributional Generalized Advantage Estimation, leveraging the Wasserstein like metric for robust policy gradient methods. DRL concepts now underpin generalizations to multi-agent and multi-objective RL. Hu et al. [14] developed a distributional reward estimation framework for more stable multi-agent learning. Hayes et al. [15] combines distributional methods with Monte-Carlo Tree Search for multi-objective problems. Eldeeb et al. [16] demonstrates the use of DRL with conservative Q-learning for offline multi-agent tasks under uncertainty.

2.2. On-policy and Off-policy Temporal Difference Updates

Deep Q-Networks (DQN) marked a breakthrough in reinforcement learning by achieving human-level performance in Atari games using raw pixels [17]. As an off-policy extension of Q-learning, DQN learns by evaluating the best theoretical next action and stabilizes training with a target network. Sarsa, by contrast, is an on-policy method that updates based on the actual action taken, making it sensitive to exploratory behavior and often yielding safer policies [18], but also creating variance in updates. Expected Sarsa sits between the two: it averages updates over all possible next actions under the current policy, reducing variance and improving stability [19]. Unlike Expected Sarsa, DQN’s maximization can cause value overestimation, which Double DQN addresses [20]. Expected Sarsa naturally mitigates this issue by averaging. [3] proves that Expected Sarsa converges under the same conditions as Sarsa and formulate specific hypotheses about when Expected Sarsa will outperform Sarsa and Q-learning. Experiments in multiple domains confirm these hypotheses and demonstrate that Expected Sarsa has significant advantages over these more commonly used methods. Despite these theoretical and empirical benefits Expected Sarsa remains underexplored in DRL.

This gap motivates our investigation into Expected Sarsa-based DRL algorithms, where the combination of expectation-based updates and distributional value representations may offer complementary advantages in stability and sample efficiency.

3. Proposed Approach: ES-C51 vs. QL-C51

This study proposes ES-C51, a modification of the standard QL-C51 algorithm. We chose C51 over other forms of DRL as the basis for our experiments as it is one of the most widely used and was compatible with the proposed Expected Sarsa modifications, moreover an easily modifiable implementation was available via CleanRL [21]. The key innovation in ES-C51 is the use of a softmax weighted expectation over next actions in the Bellman backup, rather than the greedy action used in QL-C51. This change aims to improve exploration and stabilize learning, especially in the early stages of training. In DRL the goal is to model the full distribution of returns, $Z(s, a)$ for each state action pair (s, a) , rather than just the expected return. The return distribution is represented as a vector which gives probabilities of getting each return value and the probabilities are calculated by a neural network parameterized by θ denoted by $p_\theta(s, a, z_i)$. Thus, the distribution is represented by equation 12. For action selection, both QL-C51 and ES-C51 employ a softmax policy over Q-values, parameterized by a temperature τ as shown in equation 9

$$\pi_\tau(a \mid s) = \frac{\exp(Q(s, a)/\tau)}{\sum_{b \in \mathcal{A}} \exp(Q(s, b)/\tau)} \quad (9)$$

where $\mathcal{A}$ is the set of all possible actions. The temperature τ controls the exploration exploitation trade off: high values of τ result in a nearly uniform policy (more exploration), while low values make the policy more greedy. In implementation both ES-C51 and QL-C51 use a softmax policy where an action is sampled from π_τ . The main difference between QL-C51 and ES-C51 lies in the construction of the target distribution for the temporal difference (TD) update. In QL-C51, the TD target is constructed using the distribution corresponding to the greedy action in the next state. Specifically, after computing $Q(s, a)$ for all $a \in \mathcal{A}$, the greedy action a' is selected as shown in equation 4. The distributional Bellman operator under policy π is given by equation 10

$$\mathcal{T}^\pi Z(s, a) \stackrel{D}{=} R(s, a) + \gamma \times Z(s', a'), \quad a' \sim \pi(\cdot \mid s') \quad (10)$$

where $R(s, a)$ is the immediate reward received after taking action a in state s , s' is the next state, and a' is sampled from the policy π at s' . The notation $\stackrel{D}{=}$ denotes equality in distribution. The C51 algorithm approximates the return distribution $Z(s, a)$ using a categorical distribution supported on a fixed set of N atoms $\{z_i\}_{i=0}^{N-1}$, which are uniformly spaced in the interval $[v_{\min}, v_{\max}]$. For every atom z_i in the categorical support, equation 11 computes its shifted value v_i , reflecting how the distribution’s return atoms are updated after each transition.

$$v_i = r(s, a) + \gamma z_i, \quad i \in [0, N - 1] \quad (11)$$

where $r(s, a)$ is the observed reward. Since the shifted atoms v_i will, in general, not exactly align with the fixed atom support $\{z_i\}$ the projection operator ϕ (refer to equation 7 in [6]) is applied. It redistributes the probability mass from each v_i onto the nearest atoms in the fixed support using interpolation resulting in the target distribution 12, The result is a probability vector over the original atom support forming Z_{target} for the (s, a) pair.

$$\begin{aligned} Z_{\text{target}} &= \phi\left(Z(s, a)\right) \\ Z_{\text{target}} = Z(s, a) &= \{p_\theta(s, a, z_0), p_\theta(s, a, z_1), \dots, p_\theta(s, a, z_{N-1})\} \end{aligned} \quad (12)$$

In contrast, ES-C51 constructs the TD target as the expectation over all possible next actions, weighted by their probabilities under the softmax policy shown by equation 9. The distributional Bellman operator under policy π_τ for expected sarsa update is given by Equation 13

$$\mathcal{T}^\pi Z(s, a) \stackrel{D}{=} R(s, a) + \gamma \sum_{a' \in \mathcal{A}} \pi_\tau(a'|s') \cdot Z(s', a') \quad (13)$$

This distribution is shifted using equation 11 and projected onto the fixed support using equation 12. In both algorithms, the neural network is trained by minimizing the cross entropy loss between the predicted distribution and the projected target distribution:

$$\mathcal{L}(\theta) = - \sum_{i=0}^{N-1} Z_{\text{target}}[i] \log p_\theta(s, a, z_i). \quad (14)$$

To summarize, both QL-C51 and ES-C51 share the same softmax exploration mechanism, the same projection operator ϕ , and the same cross entropy training objective. Their only difference lies in the TD backup: QL-C51

projects the distribution of the greedy action, while ES-C51 projects the expectation of the softmax weighted mixture of next action distributions as shown on Line 14 of Algorithm 1. This modification in ES-C51 is designed to reduce variance and improve stability, especially during early learning when Q-value estimates are noisy.

Table 1: Notations

Symbol	Meaning	Symbol	Meaning
s, s'	Current state, next state	a, a'	Current action, next action
$r(s, a)$	Immediate reward	γ	Discount factor
$\mathcal{A}$	Action space	$\mathcal{D}$	Replay buffer storing transitions (s, a, r, s')
N	Number of atoms	$\{z_i\}_{i=0}^{N-1}$	Fixed atom support in $[v_{\min}, v_{\max}]$
$v_{\min}, v_{\max}$	Min and max atom support values	v_i	Shifted atom value: $r(s, a) + \gamma z_i$
$p_{\theta}(s, a, z_i)$	Probability of atom z_i for (s, a)	$f_{\theta}(s, a)$	Network logits for (s, a)
p_{θ}	Neural network	p_{θ}	Target network copy of p_{θ}
$Z_{\theta}(s, a)$	Return distribution for (s, a)	$Z_{\theta'}(s, a)$	Return distribution from target network
$\bar{Z}(s')$	Expected mixture distribution	$Q(s, a)$	Expected return: $\sum_i z_i p_{\theta}(s, a, z_i)$
$\pi_{\tau}(a s)$	Softmax policy with temperature τ	τ	Temperature parameter, decaying to 0
$\mathcal{T}^{\pi}$	Distributional Bellman operator	$\mathcal{L}(\theta)$	Cross entropy loss function
Z_{target}	Target distribution after projection	θ, θ'	Network and target parameters
t, T	Current timestep, total timesteps	ϕ	Projection Operator

4. Experimental Setup and Implementation Details

All experiments were conducted on the OzSTAR supercomputing facility at Swinburne University of Technology, using NVIDIA P100 GPUs and single CPU cores. We evaluated our algorithms on Gym classic control and Atari environments, testing each game in both deterministic NoFrameskip-v4 and stochastic v0 settings. Two distributional reinforcement learning algorithms were compared: the baseline QL-C51 and our modified ES-C51. Both algorithms were set to use softmax based action selection, with QL-C51 employing a greedy Bellman backup, uses an Expected Sarsa backup with the probability of actions determined by the same softmax operation and temperature τ used for action selection as detailed in Algorithm 1. Each environment algorithm pair was trained across 10 random seeds, yielding a total of 440 independent runs. Training was managed with CleanRL, and statistical comparisons were conducted using the Wilcoxon signed rank test. Results were additionally grouped into stochastic and deterministic categories to highlight the influence of environment dynamics. Performance plots were generated by averaging across seeds, applying smoothing for clarity, and plotting mean returns with confidence intervals for both algorithms on the same axes. All

Algorithm 1 ES-C51

- 1: Initialize replay buffer $\mathcal{D}$, parameters θ of network p_θ , and target network $p_{\theta'} \leftarrow p_\theta$
 - 2: Initialize fixed support of atoms $\{z_i\}_{i=0}^{N-1}$ uniformly in $[v_{\min}, v_{\max}]$
 - 3: **for** each timestep $t = 1, \dots, T$ **do**
 - 4: Observe state s_t
 - 5: Compute logits $f_\theta(s_t, a)$ and probability masses $p_\theta(s_t, a, z_i)$ for all $a \in \mathcal{A}$ and z_i
 - 6: Compute action values $Q(s_t, a) = \sum_{i=0}^{N-1} z_i p_\theta(s_t, a, z_i)$
 - 7: Select action $a_t \sim \pi_{\tau(\cdot|s_t)}$ where
$$\pi_{\tau(a|s_t)} = \frac{\exp(Q(s_t, a)/\tau)}{\sum_{b \in \mathcal{A}} \exp(Q(s_t, b)/\tau)}$$
 - 8: Execute a_t , observe reward r_t and next state s_{t+1}
 - 9: Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
 - 10: **if** $t > \text{learning_starts}$ and $t \bmod \text{train_frequency} = 0$ **then**
 - 11: Sample mini batch (s, a, r, s') from $\mathcal{D}$
 - 12: Compute $Q(s', a') = \sum_{i=0}^{N-1} z_i p_{\theta'}(s', a', z_i)$ for all a'
 - 13: Compute $\pi_\tau(a'|s')$ as above
 - 14: Compute expected distribution for next state-action pairs:
$$\bar{Z}(s') = \sum_{a' \in \mathcal{A}} \pi_\tau(a'|s') \cdot Z(s', a')$$
 - 15: For each atom z_i from $\bar{Z}(s')$, compute shifted values $v_i = r(s, a) + \gamma z_i$ onto fixed support $\{z_i\}$ using ϕ to obtain Z_{target}
 - 16: Update θ by minimizing cross entropy loss:
$$\mathcal{L}(\theta) = - \sum_{i=0}^{N-1} Z_{\text{target}}[i] \log p_\theta(s, a, z_i)$$
 - 17: **end if**
 - 18: **if** $t \bmod \text{target_update} = 0$ **then**
 - 19: Update target network: $\theta' \leftarrow \theta$
 - 20: **end if**
 - 21: **end for**
 - 22: **end for**
-

standard hyperparameters specified in CleanRL were used for implementing both the Atari and Gym versions of ES-C51. In addition we introduced a parameter τ to enable a softmax policy. This parameter is initialized with a value of 1 and decays according to equation 15

$$\tau = \max\left(1.0 \times \left(1 - \frac{\text{Current_Timestep}}{\text{Total_Timesteps} * 0.75}\right), 0.01\right) \quad (15)$$

5. Results

5.1. Summary of results

Table 2 provides the runtime comparison of both ES-C51 and QL-C51. Despite the extra calculations required by ES-C51, for some environments it takes less execution time when trained for the same number of timesteps. ES-C51 may have more streamlined gradient updates or reduced overhead in certain parts of the learning process that offset the additional ES computations. Table 3, and Figures 3 and 4 summarise the overall performance of ES-C51 and QL-C51 across all environments. Figure 4 shows a visual analysis of improvement in every environment while Table 3 shows the exact value of mean rewards and standard deviation along with the % improvements. It considers only the final 10% of episodes from each training run, as this segment reflects the agent’s mean performance after reaching a stable, fully trained state. This approach ensures that the reported results represent the agent’s proficiency at the culmination of the learning process, minimizing the influence of early-stage variability. Pie chart 3 (a) shows that ES-C51 achieves a higher mean reward than QL-C51 on 72.7% of the environments. Box plot 3 (b) illustrates the range of improvements made by ES-C51 on stochastic and deterministic environments. Notably, deterministic environments demonstrate more consistent and superior performance with improvements ranging from -4.74% to 43.98%, with a median around 6-7% and most values concentrated between 4% and 28%. In contrast, stochastic environments show more varied and volatile results, ranging from -36.46% to 24.27%, with a median near 7% but exhibiting greater spread and several negative outliers, indicating that ES-C51’s performance is less stable in stochastic settings compared to deterministic ones.

Table 2: Runtime Comparison Analysis grouped by environment type

Env.	Winner	Imp. (%)	QL-C51 (s)	ES-C51 (s)
Gym Environments				
Acrobot-v1	ES-C51	0.4	1017.3	1013.1
CartPole-v1	ES-C51	22.5	1111.1	907.2
Atari Environments (Stochastic)				
Amidar-v0	ES-C51	4.2	64 230.9	61 669.6
BattleZone-v0	QL-C51	−6.9	67 790.8	72 839.9
Bowling-v0	QL-C51	−7.0	56 172.7	60 400.6
DoubleDunk-v0	ES-C51	13.4	68 264.6	60 206.5
Frostbite-v0	ES-C51	5.9	70 243.7	66 319.0
KungFuMaster-v0	QL-C51	−1.6	63 538.8	64 559.8
NameThisGame-v0	QL-C51	−0.7	63 246.8	63 685.4
Phoenix-v0	QL-C51	−0.2	54 065.8	54 155.7
Riverraid-v0	ES-C51	5.1	70 503.0	67 095.6
Atari Environments (Deterministic, NoFrameskip)				
AmidarNoFrameskip-v4	QL-C51	−7.4	48 386.8	52 270.9
BattleZoneNoFrameskip-v4	QL-C51	−3.0	47 953.2	49 416.5
BowlingNoFrameskip-v4	QL-C51	−3.8	46 701.2	48 525.6
DoubleDunkNoFrameskip-v4	ES-C51	5.7	54 619.1	51 685.8
FrostbiteNoFrameskip-v4	QL-C51	−0.1	50 065.2	50 114.7
KungFuMasterNoFrameskip-v4	ES-C51	9.2	50 889.1	46 618.2
NameThisGameNoFrameskip-v4	QL-C51	−1.8	52 615.5	53 604.0
PhoenixNoFrameskip-v4	ES-C51	5.7	43 458.5	41 110.9
QbertNoFrameskip-v4	ES-C51	1.6	51 539.3	50 713.8
RiverraidNoFrameskip-v4	QL-C51	−1.5	48 504.1	49 217.8

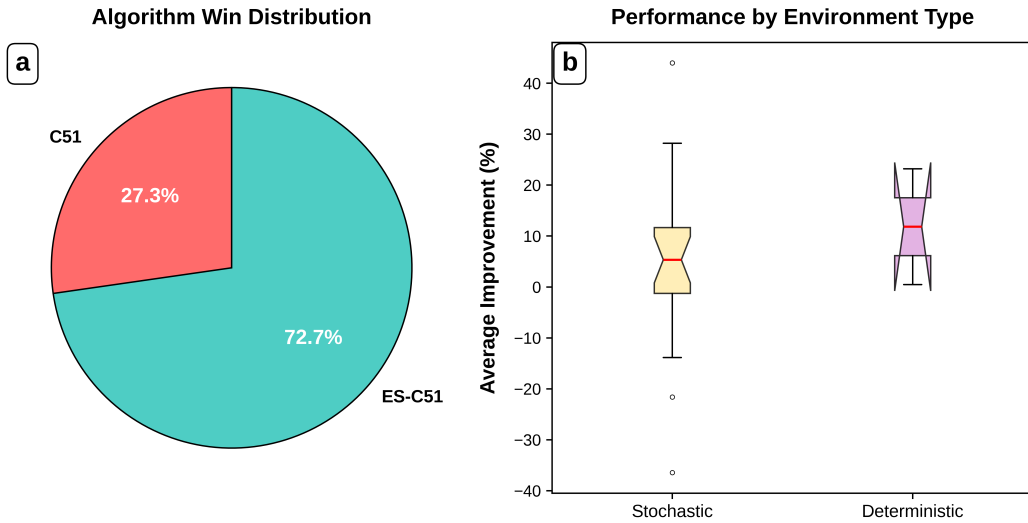


Figure 3: Performance summary

Table 3: Detailed Environment Results grouped by environment type

Env.	Winner	Imp. (%)	P-val.	Sig.	QL-C51 M $\pm$ S	ES-C51 M $\pm$ S
Gym Environments						
Acrobot-v1	ES-C51	0.47	0.5566	False	-253.63 $\pm$ 199.72	-252.45 $\pm$ 201.04
CartPole-v1	ES-C51	23.20	0.0020	True	377.07 $\pm$ 17.68	464.55 $\pm$ 12.00
Atari Stochastic Environments						
DoubleDunk-v0	ES-C51	0.40	0.0371	True	-21.20 $\pm$ 0.06	-21.12 $\pm$ 0.09
KungFuMaster-v0	ES-C51	9.68	0.1309	False	1263.45 $\pm$ 221.39	1385.74 $\pm$ 206.32
Bowling-v0	ES-C51	5.71	0.0020	True	29.06 $\pm$ 0.60	30.71 $\pm$ 0.49
Qbert-v0	QL-C51	-13.02	0.3750	False	270.57 $\pm$ 49.37	235.34 $\pm$ 104.49
Riverraid-v0	ES-C51	8.65	0.0195	True	1800.43 $\pm$ 89.73	1956.19 $\pm$ 128.39
Frostbite-v0	QL-C51	-21.60	0.0645	False	109.50 $\pm$ 16.97	85.84 $\pm$ 24.08
Phoenix-v0	ES-C51	24.27	0.0039	True	800.54 $\pm$ 66.35	994.84 $\pm$ 166.91
BattleZone-v0	QL-C51	-13.86	0.0137	True	3625.91 $\pm$ 325.37	3123.54 $\pm$ 204.38
Amidar-v0	QL-C51	-36.46	0.0098	True	19.12 $\pm$ 6.39	12.15 $\pm$ 5.18
NameThisGame-v0	ES-C51	7.95	0.1055	False	2502.24 $\pm$ 401.33	2701.29 $\pm$ 338.07
Atari Deterministic (NoFrameskip) Environments						
RiverraidNoFrameskip-v4	ES-C51	17.60	0.0020	True	1883.08 $\pm$ 123.96	2214.53 $\pm$ 95.50
NameThisGameNoFrameskip-v4	ES-C51	4.56	0.0840	False	2677.40 $\pm$ 132.45	2799.62 $\pm$ 175.13
BowlingNoFrameskip-v4	QL-C51	-4.74	0.0137	True	27.22 $\pm$ 0.67	25.93 $\pm$ 0.87
PhoenixNoFrameskip-v4	ES-C51	28.22	0.0098	True	1946.82 $\pm$ 357.10	2496.25 $\pm$ 271.67
KungFuMasterNoFrameskip-v4	ES-C51	27.56	0.0039	True	3316.21 $\pm$ 662.74	4230.29 $\pm$ 1019.56
FrostbiteNoFrameskip-v4	ES-C51	6.90	0.3750	False	109.07 $\pm$ 15.28	116.59 $\pm$ 20.42
BattleZoneNoFrameskip-v4	ES-C51	4.40	0.0840	False	2434.12 $\pm$ 137.04	2541.15 $\pm$ 149.08
AmidarNoFrameskip-v4	ES-C51	4.98	1.0000	False	3.02 $\pm$ 1.07	3.17 $\pm$ 1.72
DoubleDunkNoFrameskip-v4	QL-C51	-0.11	0.8457	False	-18.04 $\pm$ 0.21	-18.06 $\pm$ 0.17
QbertNoFrameskip-v4	ES-C51	43.98	0.9219	False	273.82 $\pm$ 117.77	394.24 $\pm$ 428.28

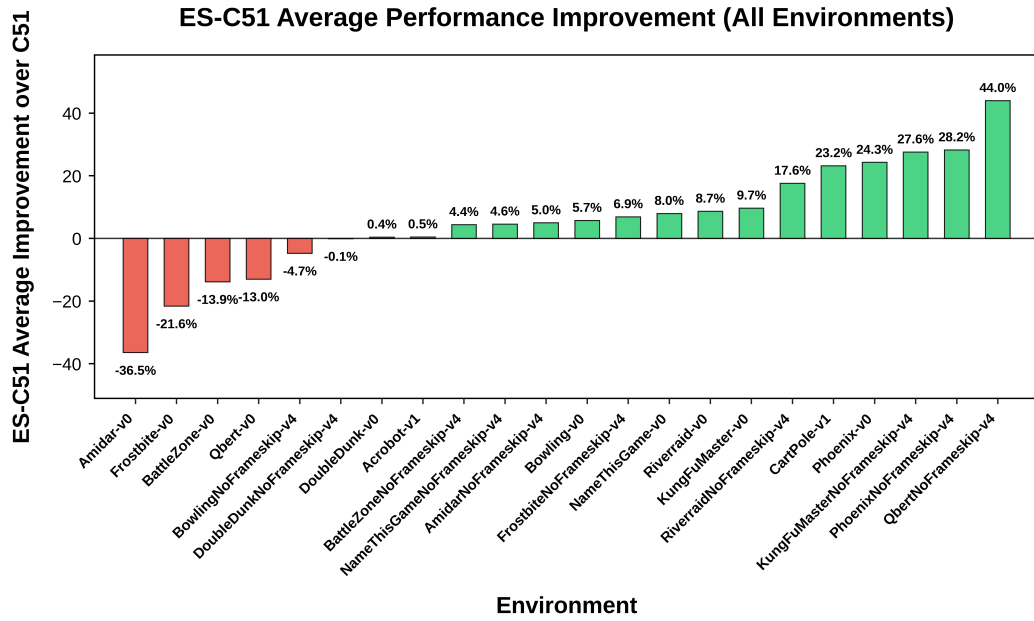


Figure 4: Average Performance Improvement by ES-C51

5.2. Analysis of Performance on Individual Environments

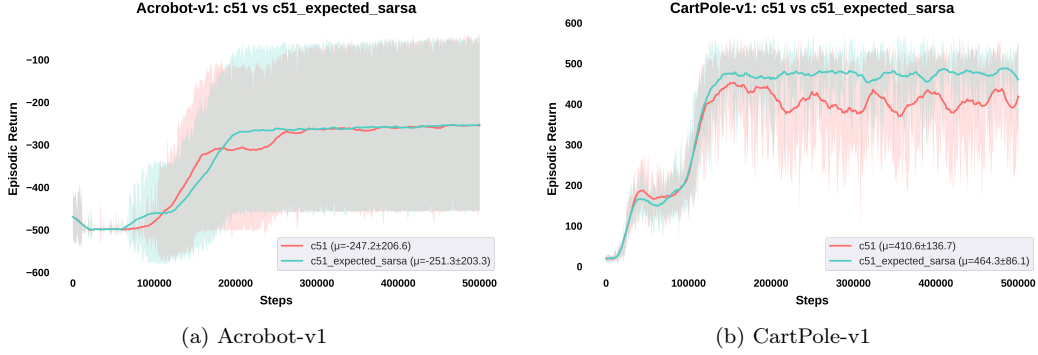


Figure 5: Performance on Gym Classic Control Environments.

Figure 5 shows the performance comparison of both algorithms on Acrobot and Cartpole classic control gym environments. Figure 5a shows that ES-C51 converged to global optimal reward at around 20000 steps while QL-C51 took 30000 steps to do the same. Moreover ES-C51 shows better stability with less fluctuations. Figure 5b shows that in Cartpole ES-C51 achieves a large jump in performance as compared to QL-C51, ES-C51 converges to a reward of around 500 while QL-C51 fluctuates around 400.

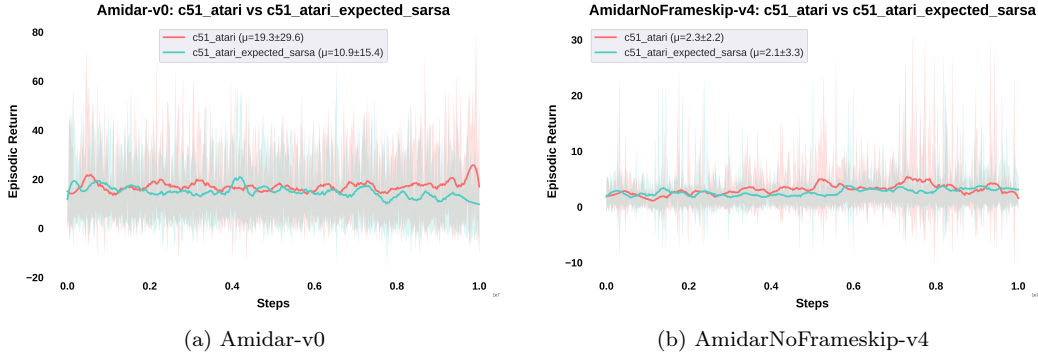


Figure 6: Comparison results for Amidar environments.

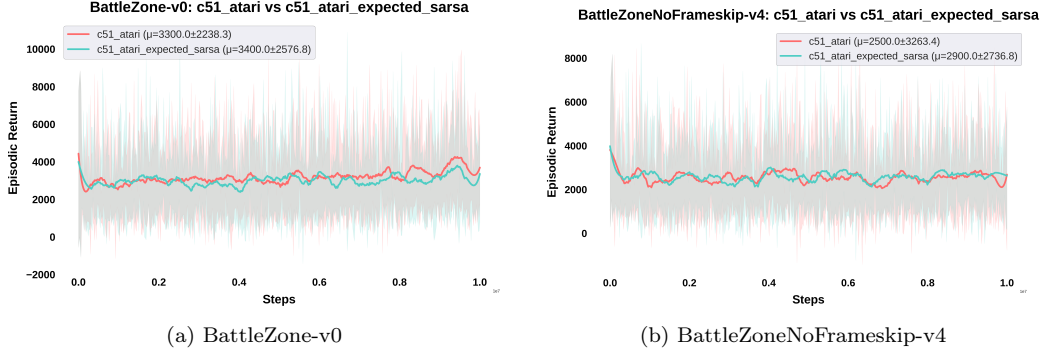


Figure 7: Comparison results for BattleZone environments.

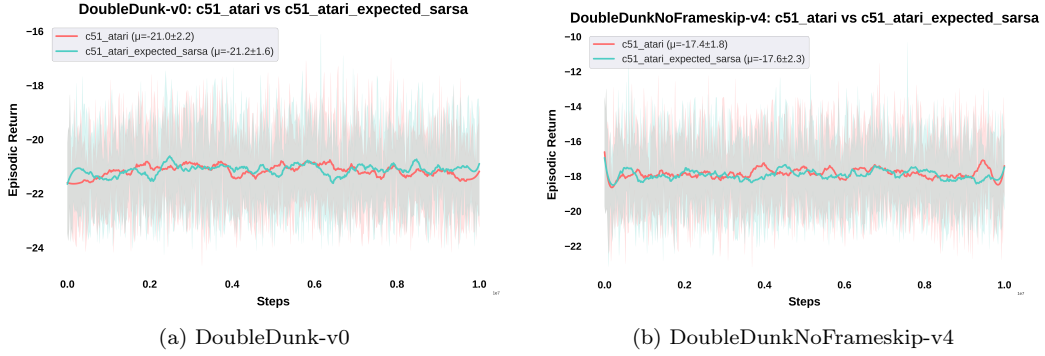


Figure 8: Comparison results for DoubleDunk environments.

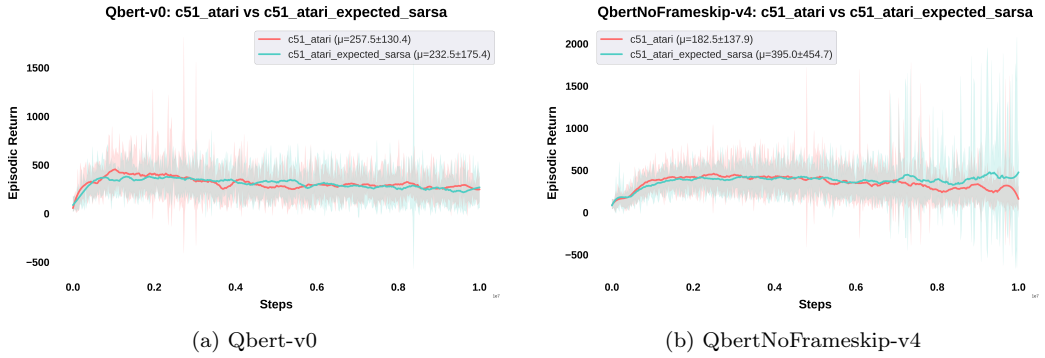


Figure 9: Comparison results for Qbert environments.

Each Atari comparison plot illustrates the performance of ES-C51 and QL-C51 on both stochastic and deterministic versions of the environments.

By convention, the NoFrameskip variant represents the deterministic setting, while the v0 variant corresponds to the stochastic setting.

Figures 6, 7, 8, and 9 present the performance comparison of QL-C51 and ES-C51 on the *Amidar*, *BattleZone*, *DoubleDunk*, and *Qbert* environments. In all cases both algorithms exhibit suboptimal performance. In *Amidar* the learning curves stabilize around an average reward of approximately 20 in the stochastic version (Figure 6a) and 5 in the deterministic version (Figure 6b). While QL-C51 slightly outperforms ES-C51 in the stochastic variant, ES-C51 achieves marginally better results in both variants. A similar trend is observed in *BattleZone*, where both algorithms plateau around an average reward of 2000 in both stochastic and deterministic versions (Figures 7a and 7b). For *DoubleDunk*, with average rewards remaining in the negative range (around -15 to -20) across both stochastic (Figure 8a) and deterministic (Figure 8b) settings, indicating that neither approach is able to effectively adapt to the environment. In contrast, the *Qbert* environment shows slightly stronger learning behavior in deterministic version (Figure 9b) where ES-C51 surpasses QL-C51, but both have similar performance in the stochastic version (Figure 9a). Overall, across this subset of environments both algorithms perform poorly, failing to learn an optimal policy.

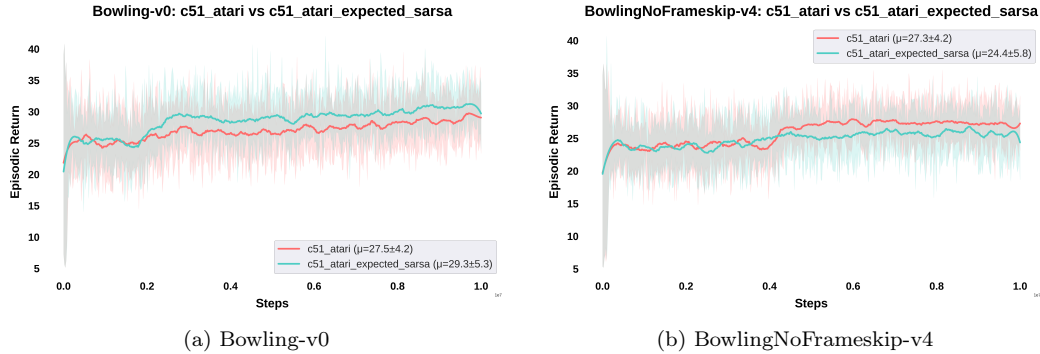


Figure 10: Comparison results for Bowling environments.

Figure 10 presents the results on the Bowling environments. In the stochastic version (10a), ES-C51 demonstrates faster convergence and achieves higher rewards, consistently surpassing QL-C51 by converging above 30, while QL-C51 struggles to reach this level. In contrast, in the deterministic version (10b), QL-C51 outperforms ES-C51, showing both higher reward convergence and more stable performance.

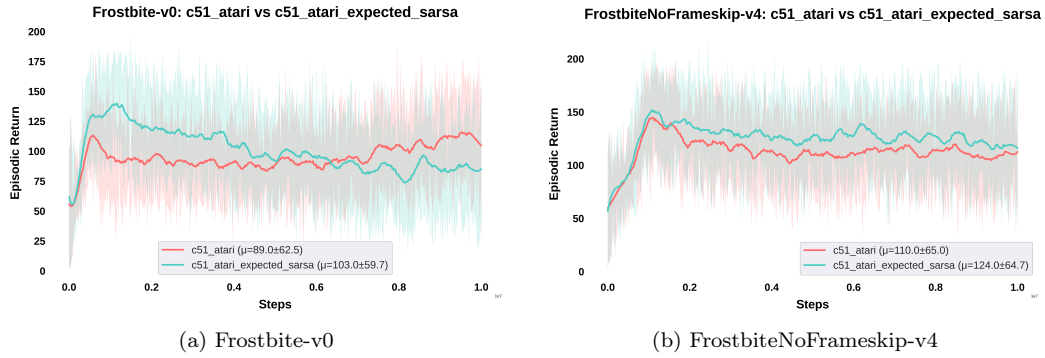


Figure 11: Comparison results for Frostbite environments.

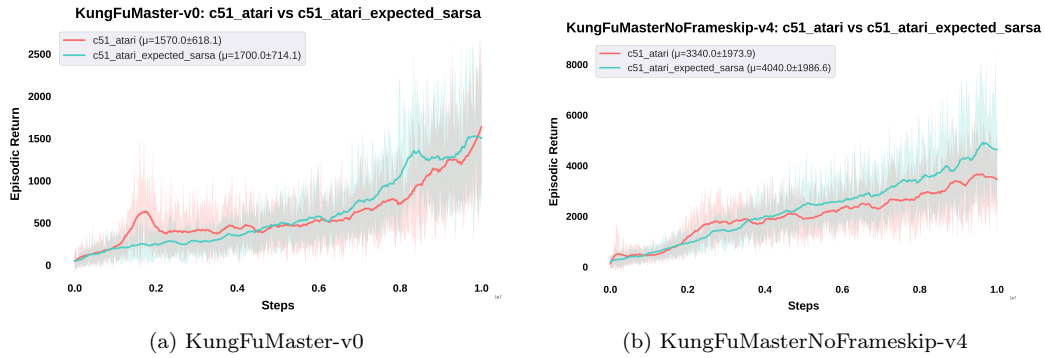


Figure 12: Comparison results for KungFuMaster environments.

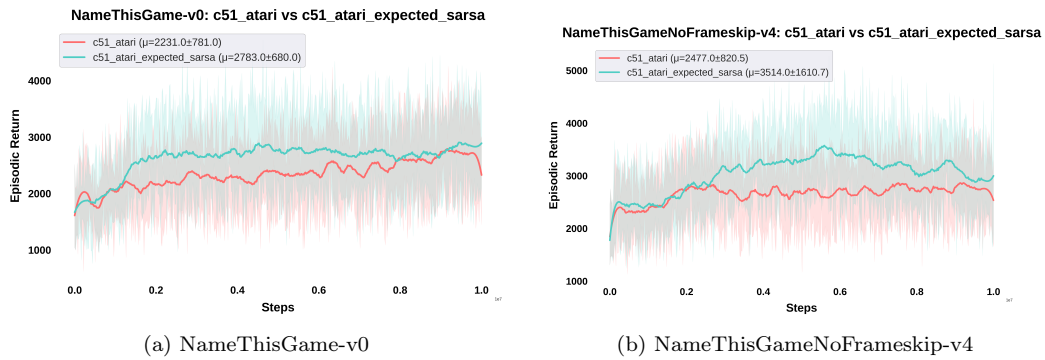


Figure 13: Comparison results for NameThisGame environments.

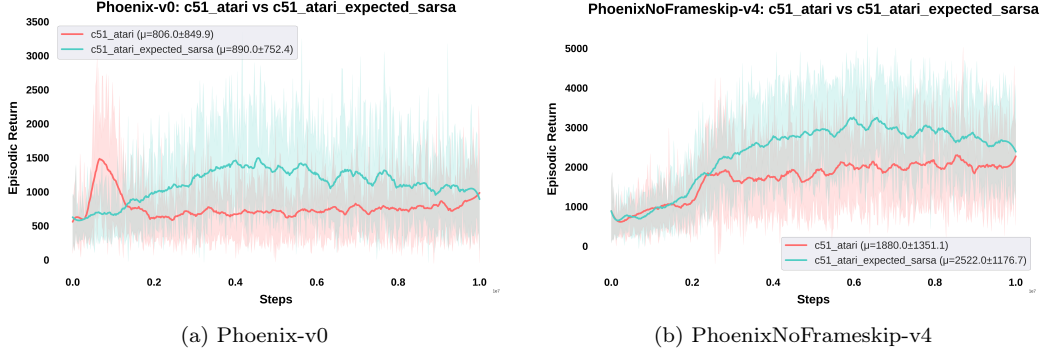


Figure 14: Comparison results for Phoenix environments.

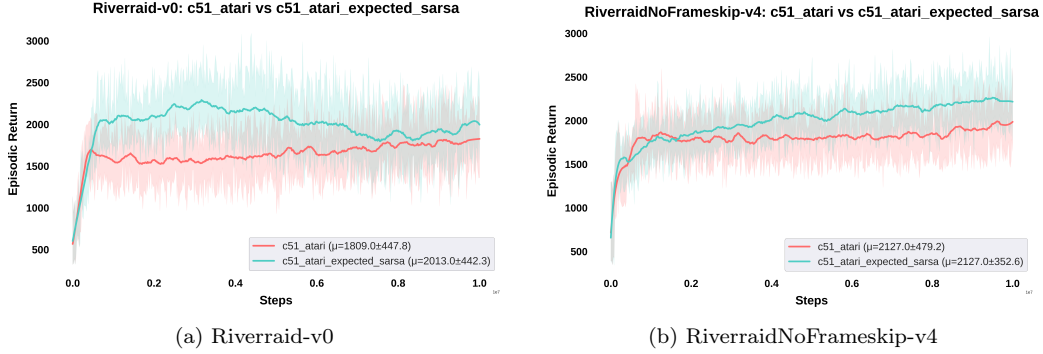


Figure 15: Comparison results for Riverraid environments.

Figures 11, 12, 13, 14, and 15 present the comparative performance of QL-C51 and ES-C51 across the *Frostbite*, *KungFuMaster*, *NameThisGame*, *Phoenix*, and *Riverraid* environments.

In the stochastic versions, a common trend is observed: ES-C51 initially achieves a sharp positive reward gain, but then drops near the end of an episode. In stochastic environments, the optimal action at a state S_t is expected to change, so greedy selection should be avoided until all time steps of an episode are completed. This behavior is managed by the parameter τ in the softmax policy.

However, this early advantage gradually decreases over time. The only notable exception is *KungFuMaster* 12a where the reward increases continuously. In general in these stochastic environments ES-C51 maintains its dominance, consistently converging to higher reward levels compared to QL-C51 (Figures 12a, 13a, 14a, 15a), With the only exception being (11a), where

ES-C51’s reward steadily declines after the initial surge, ultimately allowing QL-C51 to surpass it at around 0.7 million steps.

In the deterministic versions ES-C51 exhibits a clear and consistent advantage across all environments. Unlike in the stochastic case, its performance remains stable and continues to improve throughout training, significantly outperforming QL-C51 in *Frostbite*, *KungFuMaster*, *NameThisGame*, *Phoenix*, and *Riverraid* (Figures 11b, 12b, 13b, 14b, 15b).

We found that the performance of all algorithms is sensitive to both the value and decay of τ . Different τ settings lead to noticeable variations in performance across all environments. This indicates a broad opportunity for future work to determine the most optimal initial τ value and decay rate for each algorithm–environment pair.

Overall, ES-C51 frequently outperforms QL-C51 across both categories of environment. In the stochastic Atari environments, ES-C51 demonstrates a clear advantage, maintaining significantly higher performance up to around 0.75 million steps. Beyond this point, as τ decays to zero and the policy becomes fully greedy, ES-C51 occasionally loses its edge and experiences a performance drop in certain environments. This drawback can be alleviated by selecting the most optimal value of *tau* for that environment. Such a modification would help retain a degree of exploration and sustain ES-C51’s advantage over longer horizons. In deterministic environments, ES-C51 delivers either superior or comparable performance relative to QL-C51. Notably, in six of the evaluated environments, ES-C51 achieves substantial improvements over QL-C51, further highlighting its robustness and effectiveness.

6. Conclusion

This study addressed the limitation of greedy updates in the C51 distributional reinforcement learning algorithm and introduced a softmax based alternative ES-C51. Through extensive experimentation across a diverse set of Atari environments, results demonstrate that ES-C51 in many cases achieves superior performance compared to the baseline QL-C51 algorithm. This improvement highlights the effectiveness of softmax updates in mitigating the instability introduced by greedy policies, thereby enabling more reliable value distribution learning. Importantly these performance improvements are achieved at a computational cost which is comparable to the QL-C51 algorithm. Overall, this work establishes ES-C51 as a promising and scalable advancement and provides extensive experiments and explanations in

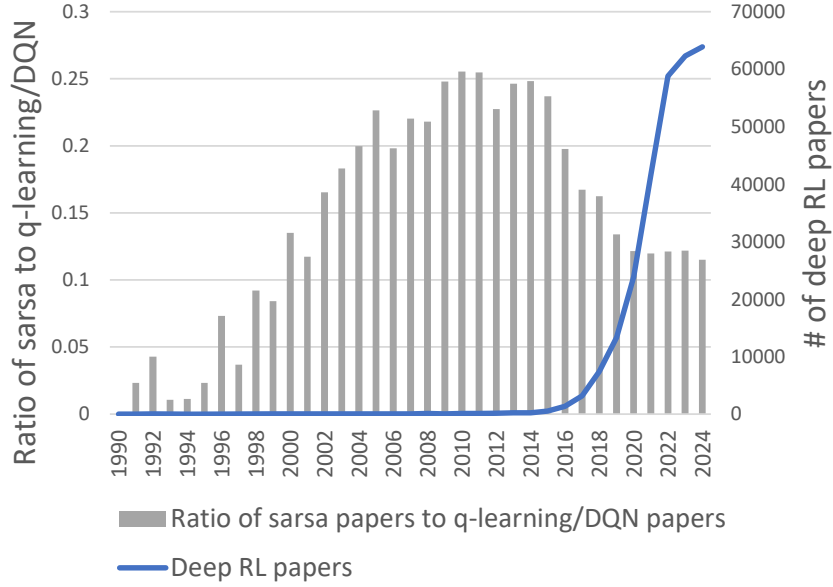


Figure 16: An analysis of publication trends in reinforcement learning, derived from Google Scholar records.

support.

More broadly we suggest our work offers a counterexample to a trend in reinforcement learning research over the last decade, which has seen diminished interest in on-policy methods such as Sarsa and Expected Sarsa relative to off-policy approaches like Q-learning. Figure 16 shows an analysis of reinforcement learning publications since 1990 (statistics derived from Google Scholar). The two decades from 1990 to 2010 saw a steady increase in the use of Sarsa compared to Q-learning - while the latter was still the dominant value-based RL approach, Sarsa had grown in popularity to be mentioned in around 25% as many papers as Q-learning. However Sarsa’s usage has waned dramatically since 2015, falling by a factor of more than two relative to Q-learning. This period coincides with the boom in deep RL. Our conjecture is that the early successes of DQN encouraged the RL research community to become overly focused on deep variants of Q-learning, largely overlooking the potential for deep extensions of Sarsa and Expected Sarsa.

The results presented in this paper provide evidence that on-policy approaches such as Expected Sarsa are compatible with deep RL, and that they may provide benefits over more widely used off-policy approaches. We hope

our work may spark renewed interest in on-policy value-based deep RL.

Data availability

The dataset of results for individual runs of each algorithm on each environment is available from https://federation.figshare.com/articles/dataset/Dataset_from_ES-C51_Expected_Sarsa_Based_C51_Distributional_Reinforcement_Learning_Algorithm/30359872, and the code implementations of ES-C51 and QL-C51 are available from https://github.com/Rijul-Tandon/cleanrl_ES-C51.git

Acknowledgements

The authors acknowledge the use of a large language model (LLM) to assist with paraphrasing and code generation for data visualization. The authors declare that this research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

- [1] P. Vamplew, C. Foale, R. Dazeley, Value function interference and greedy action selection in value-based multi-objective reinforcement learning, arXiv preprint arXiv:2402.06266 (2024).
- [2] T. Schaul, A. Barreto, J. Quan, G. Ostrovski, The phenomenon of policy churn, *Advances in Neural Information Processing Systems* 35 (2022) 2537–2549.
- [3] H. Van Seijen, H. Van Hasselt, S. Whiteson, M. Wiering, A theoretical and empirical analysis of expected sarsa, in: *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, IEEE, 2009, pp. 177–184.
- [4] M. Aitchison, P. Sweetser, M. Hutter, Atari-5: Distilling the arcade learning environment down to five games, in: *International Conference on Machine Learning*, PMLR, 2023, pp. 421–438.
- [5] M. Rowland, R. Dadashi, S. Kumar, R. Munos, M. G. Bellemare, W. Dabney, Statistics and samples in distributional reinforcement learning, in: *International Conference on Machine Learning*, PMLR, 2019, pp. 5528–5536.

- [6] M. G. Bellemare, W. Dabney, R. Munos, A distributional perspective on reinforcement learning, in: International Conference on Machine Learning, PMLR, 2017, pp. 449–458.
- [7] M. Rowland, M. Bellemare, W. Dabney, G. Ostrovski, R. Munos, C. Blundell, An analysis of categorical distributional reinforcement learning, Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) (2018) 29–37.
- [8] T. Kastner, M. Rowland, Y. Tang, M. A. Erdogdu, A.-m. Farahmand, Categorical distributional reinforcement learning with kullback-leibler divergence: Convergence and asymptotics, in: International Conference on Machine Learning, 2025.
- [9] W. Dabney, M. Rowland, M. G. Bellemare, R. Munos, Distributional reinforcement learning with quantile regression, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.
- [10] W. Dabney, G. Ostrovski, D. Silver, R. Munos, Implicit quantile networks for distributional reinforcement learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 1096–1105.
- [11] A. Jullien, H. Singh, C. Wang, C. Blundell, M. Rowland, Dual expected quantile regression for distributional rl, International Conference on Learning Representations (ICLR) (2025).
- [12] C. Enwerem, A. G. Puranic, J. S. Baras, C. Belta, Safety-aware reinforcement learning for control via risk-sensitive action-value iteration and quantile regression, 2025.
- [13] S. Shaik, J. M. Smereka, Y. Wang, Generalized advantage estimation for distributional policy gradients, in: American Control Conference, 2025.
- [14] J. Hu, Y. Sun, H. Chen, S. Huang, Y. Chang, L. Sun, et al., Distributional reward estimation for effective multi-agent deep reinforcement learning, Advances in Neural Information Processing Systems 35 (2022) 12619–12632.
- [15] C. F. Hayes, M. Reymond, D. M. Roijers, E. Howley, P. Mannion, Monte carlo tree search algorithms for risk-aware and multi-objective reinforcement learning, Autonomous Agents and Multi-Agent Systems 37 (2) (2023) 26.

- [16] E. Eldeeb, H. Sifaou, O. Simeone, M. Shehab, H. Alves, Conservative and risk-aware offline multi-agent reinforcement learning, *IEEE Transactions on Cognitive Communications and Networking* (2024).
- [17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, *nature* 518 (7540) (2015) 529–533.
- [18] G. A. Rummery, M. Niranjan, On-line Q-learning using connectionist systems, Vol. 37, University of Cambridge, Department of Engineering Cambridge, UK, 1994.
- [19] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, 2nd Edition, The MIT Press, 2018.
- [20] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30, 2016.
- [21] S. Huang, R. F. J. Dossa, C. Ye, J. Braga, D. Chakraborty, K. Mehta, J. G. AraŃsjo, Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms, *Journal of Machine Learning Research* 23 (274) (2022) 1–18.