A RELATIVE PERSPECTIVE ON DISTRIBUTIONAL REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

How much better is one action than another? We introduce Relative C51 (R-C51), a distributional reinforcement learning method that directly learns distributions of return differences between state–action pairs. A central obstacle is episode termination. When only one of two sampled transitions terminates, subtracting their Bellman targets requires absolute continuation information unavailable from relative comparisons alone. We address this with Bellman uniformization, a transformation that introduces self-loops and rescales rewards to get a common discount factor while preserving expected action values. For a fixed policy, the resulting unprojected operator is contractive, and its unique fixed point recovers the original action-value differences in expectation, although the full distributions can differ from episodic return differences. R-C51 implements this with a categorical relative critic, a learned continuation head, and an explicit mixture over successor comparisons. On the full 57-game Atari benchmark, R-C51 improves on C51 by approximately 21% in median score, 19% in interquartile mean, and 12% in optimality gap. These results support relative distributional prediction as a promising approach to value-based reinforcement learning. Source code will be made available.

1 MOTIVATION

How much better is one action than another? Value-based reinforcement learning typically answers this question by estimating each action's expected return and then comparing the predictions (Mnih et al., 2015). Could an agent learn the difference directly? Relative Value Learning recently showed that state-value differences $V^\pi(s_1) - V^\pi(s_2)$ can be learned directly with a neural network (Höftmann et al., 2026). We extend this idea to action values. The expected comparison we seek to learn is

$$\Delta^\pi\big((s_1, a_1), (s_2, a_2)\big) = Q^\pi(s_1, a_1) - Q^\pi(s_2, a_2). \tag{1}$$

This difference measures how much better one state–action pair is than another under the same policy. Choosing $s_1 = s_2$ compares two actions in the same state. Distributional reinforcement learning captures information beyond the mean by learning possible returns and their probabilities, as in C51 (Bellemare et al., 2017). Combining these ideas suggests learning a distribution over return differences. Its mean represents the Q-value difference above, while its shape describes the possible gains and losses in the comparison. We develop Relative C51 (R-C51) to learn such comparisons directly using C51's categorical representation. To handle episode termination, our construction preserves the original Q-value differences in expectation, although the full distributions can differ from episodic return differences.

Related biological findings make this combination worth exploring. Human choices and frontostriatal prediction errors depend on context and alternative options (Palminteri et al., 2015; Klein et al., 2017), while heterogeneous dopamine responses are consistent with distributional reward prediction (Dabney et al., 2020). Together, they motivate studying relative and distributional prediction, although they do not currently indicate that a neural code for return differences is present in mammalian brains. Overall, we ask whether directly learning distributional comparisons between state–action pairs can improve decision-making.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Contributions.

- **Relative Bellman formulation.** We identify why unequal transition discounts prevent ordinary relative targets and construct a Bellman uniformization that restores closure while preserving expected action values. For fixed policies, we establish Wasserstein contraction of the unprojected relative operator.
- **Relative C51.** We develop a categorical algorithm that learns comparisons directly, using a learned continuation model and mixture over successors.
- **Atari evaluation.** We evaluate R-C51 on all 57 Atari games with five seeds and compare it with C51, uniformized C51, and architecture and update-frequency ablations.

2 Related Work

Learning relative quantities connects our work to dueling networks, which separate value and advantage streams (Wang et al., 2016), and Direct Advantage Estimation, which learns advantages directly (Pan et al., 2022; Pan & Ha, 2024). Relative Value Learning predicts differences between state values (Höftmann et al., 2026) and we extend this perspective to distributions of return differences between arbitrary state–action pairs. We adopt C51's categorical representation (Bellemare et al., 2017). Closely related, Wiltzer et al. (2024) introduce distributional action gaps and superiority, a distributional analogue of advantage, for high-frequency trading. Our formulation compares independent returns and addresses unequal continuation discounts, which obstruct bootstrapping from relative information alone. Bellman uniformization restores a closed relative recursion while preserving expected action-value differences. Appendix A discusses further connections to distributional representations and modified Bellman operators.

3 Background

Basics. We consider a Markov decision process (MDP) with state space $\mathcal{S}$, action space $\mathcal{A}$, bounded rewards, and discount factor $\gamma \in [0, 1)$. Given a state–action pair $x = (s, a)$ and policy π, we denote a random successor pair as $X' = (S', A')$, where $A' \sim \pi(\cdot \mid S')$. Further, let R_x denote the random reward. In our work we deliberately distinguish the fixed discount factor γ from the random transition discount $\Gamma_x \in [0, \gamma]$ (White, 2017). In episodic tasks, $\Gamma_x = \gamma(1 - D_x)$, where $D_x = 1$ means termination. This notation makes explicit that two sampled transitions can have different continuation discounts, which matters when subtracting their Bellman targets later. Following π from x gives the return and action value

$$G_x = R_0 + \Gamma_0 R_1 + \Gamma_0 \Gamma_1 R_2 + \cdots, \qquad Q^\pi(x) := \mathbb{E}[G_x]. \tag{2}$$

Lowercase (r_x, δ_x, x') denotes a sample of (R_x, Γ_x, X'). The Bellman operator is applied on an action-value estimator q as

$$(T^\pi q)(x) = \mathbb{E}[R_x + \Gamma_x q(X') \mid x], \qquad Q^\pi = T^\pi Q^\pi. \tag{3}$$

Distributional RL. While $Q^\pi(x)$ summarizes the return by its expectation, distributional RL learns the distribution of G_x (Bellemare et al., 2017). The return decomposes into the immediate reward and the discounted continuation return,

$$G_x \overset{\mathrm{D}}{=} R_x + \Gamma_x G_{X'}, \tag{4}$$

where $\overset{\mathrm{D}}{=}$ denotes equality in distribution and $G_{X'}$ is drawn from the return distribution at the sampled successor. C51 approximates this distribution by learning probabilities on a fixed grid of return values, or atoms, $\mathcal{Z} = \{\zeta_0, \ldots, \zeta_{N-1}\}$. Its update projects the distributional Bellman target onto this grid and fits the resulting probabilities by cross-entropy (Bellemare et al., 2017; Rowland et al., 2018).

Relative Comparisons. For two state–action pairs x, y, write $\Delta^\pi(x, y) = Q^\pi(x) - Q^\pi(y)$. We compare returns from independently sampled trajectories, even when $x = y$:

$$\Delta G(x, y) := G_x - G_y, \qquad G_x \perp G_y. \tag{5}$$

Differences with a positive value favor x, and $\mathbb{E}[\Delta G(x, y)] = \Delta^\pi(x, y)$. The next section shows whether this distribution can be learned by bootstrapping from relative comparisons alone.

4 NAIVE SUBTRACTION

A relative critic should bootstrap from another return difference. For two independently sampled transitions (x, r_x, δ_x, x') and (y, r_y, δ_y, y'), subtracting their ordinary Bellman targets gives

$$Y_{xy} = \underbrace{r_x - r_y}_{\text{immediate reward diff.}} + \underbrace{\delta_x G_{x'} - \delta_y G_{y'}}_{\text{successor contribution}}. \tag{6}$$

When $\delta_x = \delta_y$, the continuation factors as $\delta_x\big(G_{x'} - G_{y'}\big)$. The update is then *closed*, since it uses the return difference represented at (x', y'). If the transition from x continues while the one from y terminates, $(\delta_x, \delta_y) = (\gamma, 0)$, then the target becomes

$$Y_{xy} = r_x - r_y + \gamma G_{x'}. \tag{7}$$

The remaining continuation term is an absolute return, but relative comparisons alone cannot identify the absolute level. This becomes evident when adding the same constant b to every continuation return. This shift leaves all their differences unchanged, but changes the target by

$$Y_{xy} \longmapsto Y_{xy} + b(\delta_x - \delta_y). \tag{8}$$

So for unequal discounts, identical relative bootstrap information can therefore correspond to different targets. In the Appendix, Theorem B.1 addresses this observation.

A zero-return absorbing reference state would restore closure, but comparisons against it are absolute return distributions. To bootstrap from relative predictions alone, we instead construct a Bellman recursion with a common discount for both branches. These design choices enable TD learning and random sampling from a replay buffer. This construction preserves the original Q-value differences in expectation, although it can change the full return-difference distributions.

5 A VALUE-PRESERVING RELATIVE BELLMAN RECURSION

For bootstrapping, we need both successors of a comparison to carry their future return with the same coefficient. A transition that terminates appears to have no future to carry, but a recursion gives us another way to represent its value. If x always terminates with reward ten, then its value is solved with both equations:

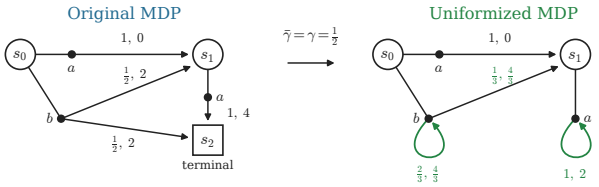

Figure 1: **Uniformized MDP.** Terminal states are removed from the MDP (White, 2017) and self-loops are added where reward and dynamics change. When $\gamma < \bar{\gamma}$, self-loops are added at *every* action node.

$$q(x) = 10 \qquad \text{and recursively} \qquad q(x) = 0.10 + 0.99q(x). \tag{9}$$

The second equation pays the reward in smaller pieces along a self-loop. It has the same solution and uses the coefficient 0.99, which looks familiar to a discount factor that a continuing branch could potentially use. Inspired by White (2017), we introduce Uniformized MDPs (see Figure 1).

Our method, that we call *Bellman uniformization*, extends this idea to stochastic transitions. We take an original Bellman target and transform it

$$\underbrace{R_x + \Gamma_x q(X')}_{\text{original target}} \quad \longrightarrow \quad \underbrace{\bar{R}_x + \bar{\gamma} q(\bar{X}')}_{\text{uniformized target}}, \tag{10}$$

so that every successor gets a *unified* coefficient $\bar{\gamma}$ independently from the termination behavior expressed by Γ_x. Hereby, quantities marked with a bar $(\bar{R}_x, \bar{X}')$ refer to objects after the transformation. In fact, the reward $\bar{R}_x$ is rescaled, and the successor $\bar{X}'$ either stays at x or advances to the sampled successor X'. As in classical uniformization, self-loops absorb local variation so that one common coefficient can be used throughout (Lippman, 1975; Puterman, 2014).

Next, we identify how the reward and successor weights need to be adjusted. The transformed target may differ from the original, but its expected update has the same value function as its fixed point.

5.1 PRESERVING VALUES WITH SELF-LOOPS

A self-loop keeps some weight at x. To introduce that weight without changing the fixed point, take only a fraction $\rho_x > 0$ of the Bellman update using its residual:

$$q(x) + \rho_x\big((T^\pi q)(x) - q(x)\big) = (1-\rho_x)q(x) + \rho_x \mathbb{E}[R_x + \Gamma_x q(X') \mid x]. \tag{11}$$

The Bellman residual vanishes at every x exactly when $q = Q^\pi$, and multiplying it by a positive factor at each x therefore preserves the fixed point. Thus ρ_x lets us trade weight on the original target for weight on $q(x)$. Although ρ_x resembles a learning rate or EMA parameter, here it defines the transformed Bellman operator. Once the common discount is chosen, the environment dynamics determine ρ_x alone.

The update preserves the fixed point, but we want to interpret the update as taking one step with discount $\bar{\gamma}$. That step can either stay at x or advance to a successor. Since these possibilities exhaust the next step, their probabilities must sum to one, so their discounted weights must sum to $\bar{\gamma}$. To determine them, introduce

$$c_x := \mathbb{E}[\Gamma_x \mid x] \in [0, \gamma], \tag{12}$$

as the *expected transition discount*. For binary termination, we get $c_x = \gamma \Pr(D_x = 0 \mid x)$. Note that c_x measures expected discounted continuation, rather than continuation probability itself. We use c_x to calculate the total successor weight, while the dependence between Γ_x and X' remains inside the Bellman expectation.

Choose the shared discount coefficient $\bar{\gamma} \in [\gamma, 1)$, with $\bar{\gamma} > 0$. The lower bound will ensure valid successor probabilities, while the upper bound gives a discounted, contractive Bellman operator. In the expected update, the weight at x is $1-\rho_x$, and the total weight on original successors is $\rho_x c_x$. Matching their sum to the shared discount gives

$$\underbrace{1-\rho_x}_{\text{weight at } x} + \underbrace{\rho_x c_x}_{\text{weight on original successors}} = \bar{\gamma} \quad \Longrightarrow \quad \rho_x = \frac{1-\bar{\gamma}}{1-c_x}. \tag{13}$$

Since $c_x \le \gamma \le \bar{\gamma} < 1$, we obtain $0 < \rho_x \le 1$. In Equation 11, the same factor scales the reward, giving $\bar{r}_x = \rho_x r_x$ for each sample. These weights specify the update in expectation. We next construct an update rule that ensures these weights are realized using sampled transitions.

5.2 STAY OR GO

The previous subsection determined the expected weights. We now realize them through a constructed transition. Given an environment sample $\tau_x = (x, r_x, \delta_x, x')$, it can either stay at x or advance to x'. The two target options are:

$$\underbrace{\bar{r}_x + \bar{\gamma} q(x)}_{\text{stay at } x} \qquad \text{or} \qquad \underbrace{\bar{r}_x + \bar{\gamma} q(x')}_{\text{go to } x'}. \tag{14}$$

Both use the same discount, as required for a relative backup. We need to choose how often each target is used so that their average recovers the value-preserving update from Section 5.1. Equation 13 describes how often one should select x or a real successor x' by rearranging it: $\frac{1-\rho_x}{\bar{\gamma}} + \frac{\rho_x c_x}{\bar{\gamma}} = 1$ provides the overall stay and advance probabilities. Therefore, we write the expected update as

$$\mathbb{E}[\underbrace{\bar{R}_x + \bar{\gamma} q(\bar{X}')}_{\text{uniformized target}}] = \rho_x \mathbb{E}[R_x \mid x] + \bar{\gamma}\left[\frac{1-\rho_x}{\bar{\gamma}} q(x) + \frac{\rho_x}{\bar{\gamma}} \mathbb{E}\left[\Gamma_x q(X') \mid x\right]\right]. \tag{15}$$

The overall probability of advancing is $\frac{\rho_x c_x}{\bar{\gamma}}$, but it does not yet specify which transitions should lead to an advance. The discount and successor can be dependent, so their joint contribution must be preserved. In particular, a terminal sample must contribute no bootstrap term from its original successor. We therefore condition the advance probability on the sampled discount δ_x. The original

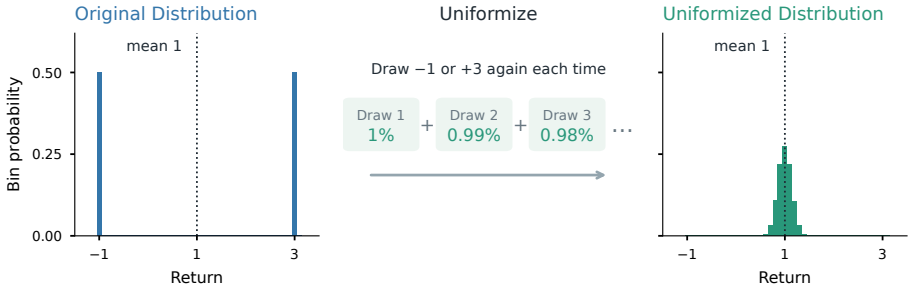

Figure 2: **Uniformization Preserves the Mean.** A terminal reward takes values -1 and $+3$ with equal probability. With discount $\gamma = 0.99$, uniformization scales each independently sampled reward by $1 - \gamma = 0.01$. Both returns have mean 1, but their distributions differ.

successor contribution is $\rho_x \delta_x q(x')$. Advancing with probability g_x under the common discount contributes $g_x \bar{\gamma} q(x')$ on average. Matching these gives

$$g_x \bar{\gamma} = \rho_x \delta_x \implies g_x = \frac{\rho_x \delta_x}{\bar{\gamma}}, \tag{16}$$

with $0 \le \rho_x \delta_x \le \gamma \le \bar{\gamma}$ and therefore $g_x \in [0, 1]$. Thus terminal samples never advance, while continuing samples advance often enough to keep their required successor contribution. Since $\mathbb{E}[\delta_x \mid x] = c_x$, this rule also recovers the overall advance probability.

Conditional on the sampled transition τ_x, the constructed successor is

$$\bar{X}' = \begin{cases} x', & \text{with probability } g_x, \\ x, & \text{with probability } 1 - g_x. \end{cases} \tag{17}$$

While this defines a random process, the Relative C51 algorithm avoids this sampling choice by evaluating the successor mixture directly.

It remains to check the contribution from staying. Conditional on the sampled transition, the stay probability is $1 - g_x$, giving weight $\bar{\gamma}(1 - g_x) = \bar{\gamma} - \rho_x \delta_x$ to $q(x)$. Together with the reward and advance contribution, this gives

$$\mathbb{E}\big[\bar{r}_x + \bar{\gamma} q(\bar{X}') \mid \tau_x\big] = \rho_x r_x + (\bar{\gamma} - \rho_x \delta_x) q(x) + \rho_x \delta_x q(x'). \tag{18}$$

The stay weight varies with the sampled discount, but its average is exactly the weight derived in Section 5.1,

$$\mathbb{E}[\bar{\gamma} - \rho_x \Gamma_x \mid x] = \bar{\gamma} - \rho_x c_x = 1 - \rho_x. \tag{19}$$

Averaging the full target over sampled environment transitions therefore gives

$$\mathbb{E}[\bar{R}_x + \bar{\gamma} q(\bar{X}') \mid x] = (1 - \rho_x) q(x) + \rho_x (T^\pi q)(x). \tag{20}$$

Thus the constructed step realizes the value-preserving update while using the same discount $\bar{\gamma}$ on every step. In the Appendix, Proposition B.2 shows the formal value-preservation and contraction guarantees.

5.3 Closing the Relative Recursion

Following the stay-or-go rule defines the new discounted return

$$H_x := \sum_{t \ge 0} \bar{\gamma}^t \bar{R}_t, \qquad H_x \overset{\mathrm{D}}{=} \bar{R}_x + \bar{\gamma} H_{\bar{X}'}. \tag{21}$$

Table 1: **Successor Pairs and Weights.** Given two replay transitions, the branches independently advance with probabilities g_x and g_y, and otherwise stay. Each row contributes the return-difference distribution at its successor pair, weighted by the probability of that outcome.

Branch x	**Branch** y	**Successor comparison**	**Mixture weight**
Stay	Stay	(x, y)	$w_{00} = (1 - g_x)(1 - g_y)$
Stay	Go	(x, y')	$w_{01} = (1 - g_x)g_y$
Go	Stay	(x', y)	$w_{10} = g_x(1 - g_y)$
Go	Go	(x', y')	$w_{11} = g_x g_y$

As for G_x, we leave the policy implicit. For two state–action pairs x, y, define their return difference

$$\Delta H(x, y) := H_x - H_y, \qquad H_x \perp H_y. \tag{22}$$

Finally, both successor returns carry the same coefficient $\bar{\gamma}$ and subtracting their recursions therefore lets us factor out this discount and form another return difference,

$$\Delta H(x, y) \overset{\mathrm{D}}{=} \bar{R}_x - \bar{R}_y + \bar{\gamma}\Delta H(\bar{X}', \bar{Y}'). \tag{23}$$

The recursion remains closed even when only one environment transition terminates because that branch stays at its current pair and keeps the common discount. Then, value preservation gives

$$\mathbb{E}[\Delta H(x, y)] = \Delta^\pi(x, y) = Q^\pi(x) - Q^\pi(y), \tag{24}$$

although the full distribution can differ from the original return difference. In the Appendix, Theorem B.3 establishes fixed-policy Wasserstein contraction and identifies the unique fixed point. The next section evaluates the four possible successor comparisons as a mixture of categorical distributions.

6 Relative C51

R-C51 approximates the uniformized return-difference distribution $\Delta H(x, y)$ by a categorical distribution $p_\theta(x, y)$ on fixed atoms $\mathcal{Z} = \{\zeta_0, \ldots, \zeta_{N-1}\}$. To construct a training target, we estimate the continuation discounts, combine the four possible successor distributions, and project the resulting Bellman target onto these atoms.

Our implementation uses $\bar{\gamma} = \gamma$, which minimizes the probability of introduced self-loops. When continuation is certain ($c_x = \gamma$), it gives $\rho_x = g_x = 1$ and recovers the original reward and transition without introducing any self-loop.

6.1 The four-way target mixture

To construct a training target for $p_\theta(x, y)$, we combine all possible successor comparisons under the stay-or-go rule. Each branch either stays at the current state–action pair or goes to its sampled successor, resulting in four outcomes (see Table 1). We evaluate all four successor distributions and weight them by their probabilities, rather than sampling only a single outcome (as described in Equation 17).

Draw two transitions independently from the replay buffer,

$$\tau_x = (x, r_x, \delta_x, s'_x), \qquad \tau_y = (y, r_y, \delta_y, s'_y). \tag{25}$$

For episodic tasks, a continuation head predicts the probability of nontermination. Targets use the target network's continuation estimates and the online network is trained by binary cross-entropy against the observed nontermination indicators $1 - d_x$. Multiplying the head's prediction by γ gives the estimated expected transition discount $\hat{c}_x$. Then, the reward scale and go probability can be computed for both transitions $u \in \{x, y\}$ as

$$\rho_u = \frac{1 - \bar{\gamma}}{1 - \widehat{c}_u}, \qquad \bar{r}_u = \rho_u r_u, \qquad g_u = \frac{\rho_u \delta_u}{\bar{\gamma}}. \tag{26}$$

In the following, x' and y' denote the observed successor state *paired* with the target critic's greedy action (see Section 6.2 on how to select it). On termination, $g_u = 0$, we set $u' = u$ because no successor action is needed.

The two branches independently stay or advance, giving the four successor comparisons and product weights in Table 1. Evaluating these pairs with the target critic p_{θ^-} and combining their probability vectors gives

$$\nu = w_{00}p_{\theta^-}(x, y) + w_{01}p_{\theta^-}(x, y') + w_{10}p_{\theta^-}(x', y) + w_{11}p_{\theta^-}(x', y'), \tag{27}$$

where ν_k is the probability assigned to atom ζ_k by the mixture. To obtain the Bellman target, we shift and scale each atom while keeping its probability:

$$v_k := \bar{r}_x - \bar{r}_y + \bar{\gamma}\zeta_k. \tag{28}$$

Applying the standard C51 projection (Rowland et al., 2018) to the transformed atoms v_k with probabilities ν_k gives the target histogram m. We train the online critic as

$$\mathcal{L}_{\mathrm{RC51}}(x, y) = -\sum_{k=0}^{N-1} m_k \log p_{\theta,k}(x, y). \tag{29}$$

We jointly optimize the distributional loss and continuation loss. Their weighting and the target-network update schedule are given in Appendix C.

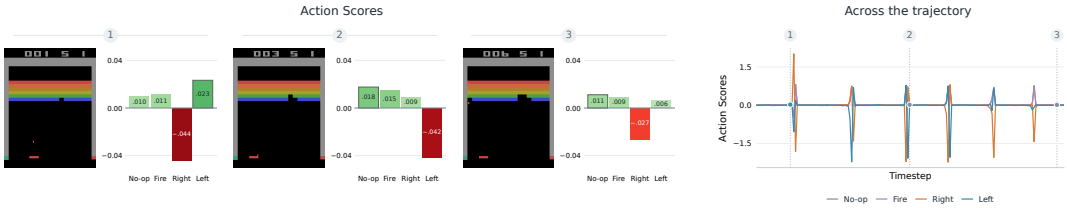

Figure 3: **Action Selection in Breakout.** Left: Three states from a trajectory and their action scores $S_\theta(s, a)$ (see Equation 31). The outlined bar indicates the greedy action. Right: Scores for all four actions along the trajectory, with numbered markers identifying the three displayed states. Higher scores indicate preferred actions.

6.2 Action Selection

Action selection requires only a ranking of actions. Action-value differences against any common reference can compute this ranking. We use every action in the current state as a reference and combine predictions in both comparison directions. Let $x_a = (s, a)$, let A be the number of actions, and write

$$\mu_\theta(x, y) := \sum_k \zeta_k p_{\theta,k}(x, y) \tag{30}$$

for the predicted mean difference. Both $\mu_\theta(x_a, x_b)$ and $-\mu_\theta(x_b, x_a)$ estimate $Q^\pi(s, a) - Q^\pi(s, b)$, but these estimates might not agree for the learned critic. We average them and then average over reference actions to get the action score

$$S_\theta(s, a) = \frac{1}{2A}\sum_{b=1}^{A}\big[\mu_\theta(x_a, x_b) - \mu_\theta(x_b, x_a)\big]. \tag{31}$$

This aggregation is the least-squares reconstruction of centered action scores from antisymmetrized pairwise comparisons, a special case of Jiang et al. (2011). An action with the largest score is the greedy action. If the predicted means satisfy $\mu_\theta(x_a, x_b) = Q^\pi(s, a) - Q^\pi(s, b)$ for every action pair a, b, then Equation 31 reduces to

$$S_\theta(s, a) = Q^\pi(s, a) - \frac{1}{A}\sum_{b=1}^{A} Q^\pi(s, b). \tag{32}$$

The subtracted baseline is independent of a, so maximizing this score is equivalent to maximizing $Q^\pi(s, a)$. The online network uses this score for its ϵ-greedy policy. For training targets, the target network both selects greedy successor actions and evaluates their distributions, following DQN (Mnih et al., 2015).

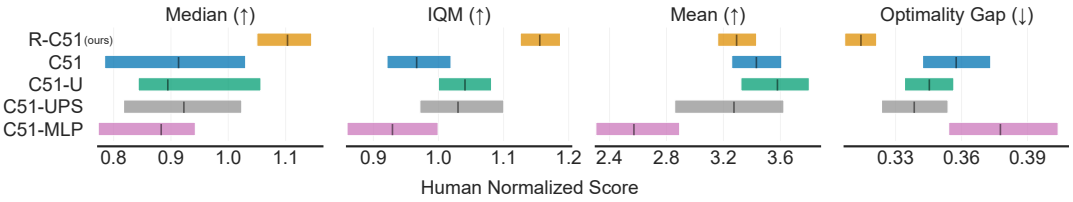

Figure 4: **Aggregated Metrics for R-C51, C51, Uniformized C51 and Ablations.** Aggregate metrics with 95% stratified bootstrap confidence intervals (Agarwal et al., 2021). Higher median, interquartile mean (IQM), and mean, but lower optimality gap indicate better performance.

7 Experiments

We evaluate whether learning return-difference distributions directly improves Atari performance over C51, and whether the gains can be explained by network capacity, update frequency, or Bellman uniformization alone.

Evaluation protocol. We evaluate R-C51 on all 57 Atari games using five independent seeds (Bellemare et al., 2013). Each run is trained for 10M steps (40M frames) in total, and the evaluation averages the returns from 100 full-game episodes with ϵ-greedy exploration at $\epsilon = 0.001$. We report the median, interquartile mean (IQM), mean, and optimality gap, with 95% stratified bootstrap confidence intervals (Agarwal et al., 2021) and single game performance in the Appendix (see Table 3). Our implementation uses two parallel EnvPool environments (Weng et al., 2022). Further settings are provided in Appendix C which follow the DQN / C51 training protocol.

The C51 baseline is from Google DeepMind's DQN Zoo[1] (Quan & Ostrovski, 2020) and we use evaluation results at their 40M-training-frame checkpoint, matching our training budget. For the ablations, we use our own EnvPool-based C51 implementation, which achieves similar aggregate scores. This allows us to compare the ablations within a shared implementation. We use five seeds for uniformized C51 (C51-U) and three seeds each for the capacity and update-frequency ablations.

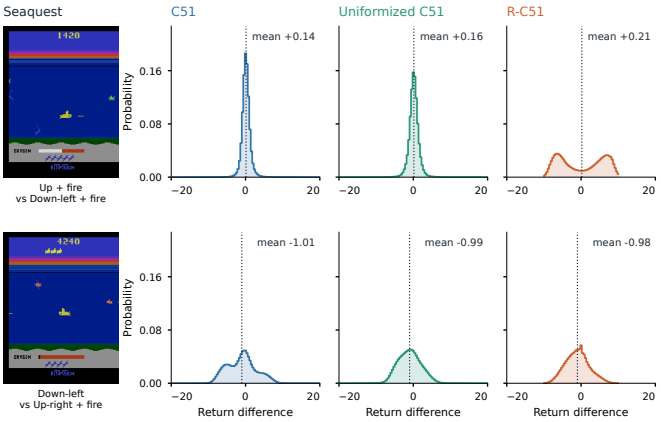

Figure 5: **Return-difference Distributions.** Each row compares two actions in the displayed state. C51 and uniformized C51 derive the difference distribution from their predicted return distributions, whereas R-C51 predicts it directly. The distributions have similar means (dotted lines) but can differ in shape.

Atari performance. Figure 4 summarizes performance across the Atari suite. R-C51 improves over C51 the median score from 0.913 to 1.103 and the IQM from 0.966 to 1.155, corresponding to gains of approximately 21% and 19%, respectively. It also reduces the optimality gap from 0.357 to 0.314, a reduction of approximately 12%. The mean score is slightly lower (3.292 versus 3.431). The results for individual games are reported in Table 3.

Ablations. We first ablate whether additional network capacity or more frequent optimizer updates can account for R-C51's performance. C51-MLP adds two 512-unit layers after the CNN encoder

[1]Code and experiment data: https://github.com/google-deepmind/dqn_zoo

to test whether additional network capacity can explain R-C51's performance gains. This increases C51's network capacity, although one could argue that R-C51's MLP has a different functionality than increasing network capacity. In addition, the second ablation C51-UPS matches R-C51's number of optimizer updates. The results are that C51-MLP performs worse, particularly bad in mean score, while C51-UPS achieves a similar median to C51, a lower mean, and modest improvements in IQM and optimality gap. Neither modification closes the observed performance gap to R-C51.

To understand the contribution of uniformization, we evaluate C51-U. An absolute categorical critic is trained on the uniformized MDP, with a learned continuation model and the same update frequency as R-C51. C51-U improves over the baseline and performs similarly to C51-UPS, but remains below R-C51 on the reported aggregate metrics except mean. This comparison supports the benefit of learning relative distributions beyond applying uniformization to an absolute critic.

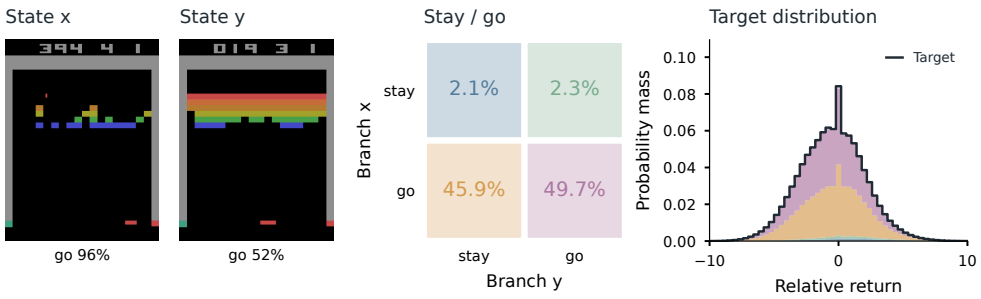

Figure 6: **Weighted Target Mixture.** Left: Two states with their predicted go probabilities. Center: Weights of the four stay-or-go combinations for each pair. Right: Their color-matched contributions to the target distribution of relative returns. The black outline shows the combined target.

Qualitative analysis. Figure 5 compares predicted return-difference distributions for actions in a Seaquest state. For C51 and C51-U, we construct these distributions by subtracting independent returns drawn from their predicted absolute distributions. R-C51 outputs the comparison directly. The shown distributions have similar means but potentially different shapes. Exact mean preservation is guaranteed by the uniformized process. Not only can we compare actions within a state. R-C51 also enables to compare arbitrary combinations of state–action pairs which is visualized in Figure 7.

Figure 6 illustrates the target construction. The continuation predictions determine the weights of the four stay-or-go pairs, whose categorical distributions combine into the target mixture. When one transition terminates, its branch stays while the other can advance, allowing the update to continue using relative predictions.

Limitations. Our guarantees apply to the unprojected fixed-policy operator with exact continuation expectations. Learned continuation estimates and categorical clipping can introduce bias, and the analysis does not prove convergence under neural function approximation and greedy control. Uniformization preserves expected values but changes return distributions, thereby limiting the interpretation as episodic outcome uncertainty. Empirical performance is evaluated on Atari and comes with approximately 50% longer training time than C51.

8 Conclusion

We introduced Relative C51, which directly learns distributional comparisons between state–action pairs. Bellman uniformization handles unequal termination by preserving expected action values under a common discount, although it can change the full return-difference distributions. The unprojected relative operator is contractive for fixed policies. Across 57 Atari games, R-C51 improves on C51 by approximately 21% in median score and 19% in interquartile mean, while reducing the optimality gap by 12%. These results support relative distributional prediction as a promising approach to value-based reinforcement learning.

AI USE STATEMENT

In this work, we used generative AI tools for aid and polish writing, for retrieval and discovery, for research execution (implement methods), to draft sections of the paper and assist in the writing of proofs.

We have not used generative AI tools for generating synthetic datasets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis or interpret results.

Additionally, we used generative AI tools for some figure generation. We have reviewed all AI-assisted work. We checked LLM-generated figures for their correct data usage. LLM-generated code was verified and tested (with LLM-generated tests) for correctness by 1 author. Human-generated research ideas were executed by LLM agents. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47: 253–279, 2013.

Marc G Bellemare, Georg Ostrovski, Arthur Guez, Philip Thomas, and Rémi Munos. Increasing the action gap: New operators for reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pp. 449–458. Pmlr, 2017.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pp. 1096–1105. PMLR, 2018a.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018b.

Will Dabney, Zeb Kurth-Nelson, Naoshige Uchida, Clara Kwon Starkweather, Demis Hassabis, Rémi Munos, and Matthew Botvinick. A distributional code for value in dopamine-based reinforcement learning. *Nature*, 577(7792):671–675, 2020.

Marc Höftmann, Jan Robine, and Stefan Harmeling. Relative value learning. *arXiv preprint arXiv:2607.21120*, 2026.

Xiaoye Jiang, Lek-Heng Lim, Yuan Yao, and Yinyu Ye. Statistical ranking and combinatorial hodge theory. *Mathematical Programming*, 127(1):203–244, 2011.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tilmann A Klein, Markus Ullsperger, and Gerhard Jocham. Learning relative values in the striatum induces violations of normative decision making. *Nature communications*, 8(1):16033, 2017.

Steven A Lippman. Applying a new device in the optimization of exponential queuing systems. *Operations research*, 23(4):687–710, 1975.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

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.

Stefano Palminteri, Mehdi Khamassi, Mateus Joffily, and Giorgio Coricelli. Contextual modulation of value signals in reward and punishment learning. *Nature communications*, 6(1):8096, 2015.

Hsiao-Ru Pan and David Ha. Skill or luck? return decomposition via advantage functions. In *International Conference on Learning Representations*, volume 2024, pp. 9282–9304, 2024.

Hsiao-Ru Pan, Nico Gürtler, Alexander Neitz, and Bernhard Schölkopf. Direct advantage estimation. *Advances in Neural Information Processing Systems*, 35:11869–11880, 2022.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

John Quan and Georg Ostrovski. DQN Zoo: Reference implementations of DQN-based agents, 2020. URL http://github.com/deepmind/dqn_zoo.

Mark Rowland, Marc Bellemare, Will Dabney, Rémi Munos, and Yee Whye Teh. An analysis of categorical distributional reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 29–37. PMLR, 2018.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, et al. Envpool: A highly parallel reinforcement learning environment execution engine. *Advances in Neural Information Processing Systems*, 35:22409–22421, 2022.

Martha White. Unifying task specification in reinforcement learning. In *International Conference on Machine Learning*, pp. 3742–3750. PMLR, 2017.

Harley Wiltzer, Marc G Bellemare, David Meger, Patrick Shafto, and Yash Jhaveri. Action gaps and advantages in continuous-time distributional reinforcement learning. *Advances in Neural Information Processing Systems*, 37:47815–47848, 2024.

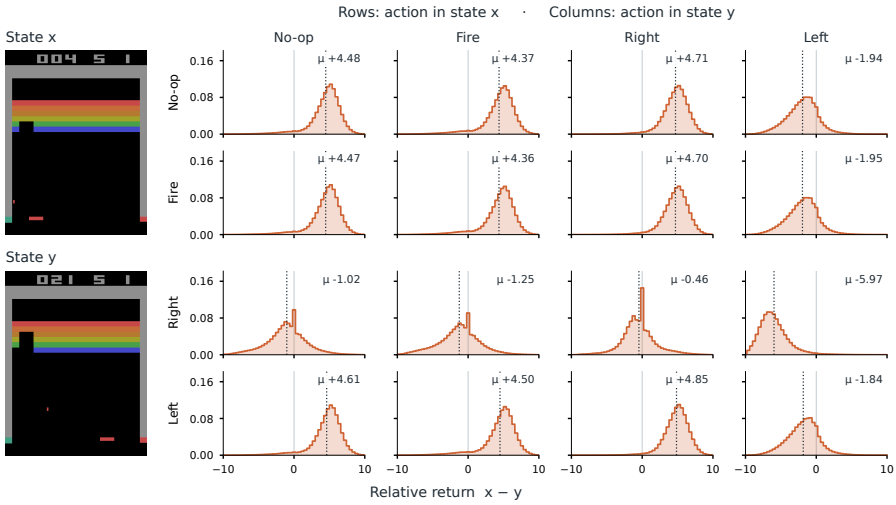

Figure 7: **Pairwise Return-difference Distributions in Breakout.** R-C51 predictions for all 16 action pairs between the two displayed states. Rows correspond to actions in state x and columns to actions in state y. Each panel shows the predicted distribution of return differences $(x - y)$, with its mean marked by a dotted line. Positive means favor the state–action pair in x.

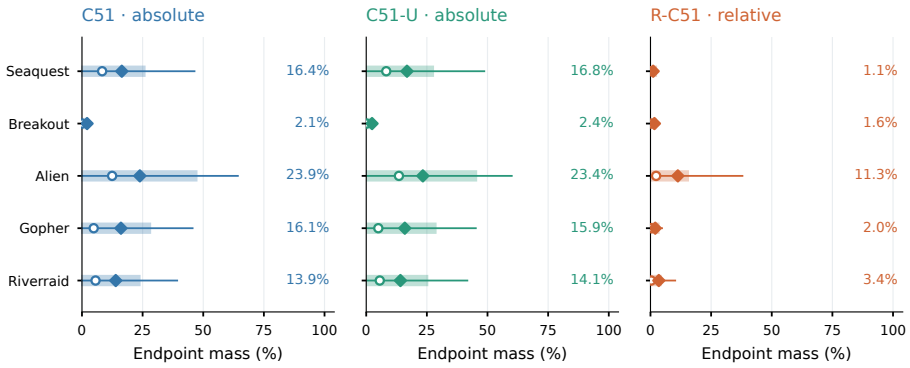

Figure 8: **Endpoint Probability Mass on** $[-10, 10]$. Given are five Atari games for C51 (blue), C51-U (green), and R-C51 (orange), using a shared observation dataset. C51 and C51-U average endpoint mass over all actions within each state. R-C51 pairs each observation with 12 randomly sampled other observations and averages over all (ordered) action combinations for each state pair. Identical observations are excluded. Diamonds show means, circles show medians, and thick/thin bars span the 25th–75th / 10th–90th percentiles. Because the methods represent different return variables, endpoint mass presents pressure on the support range. R-C51 shows lower endpoint mass on the sampled comparisons, suggesting less support saturation. Also, endpoint mass serves as an imperfect proxy for clipping-induced value bias but does not directly quantify it.

A Additional Related Work

Distributional representations. Distributional reinforcement learning models the distribution of returns rather than only its expectation. Beyond the fixed categorical support used by C51 (Bellemare et al., 2017), QR-DQN learns quantile locations (Dabney et al., 2018b), and implicit quantile networks approximate the quantile function (Dabney et al., 2018a). These approaches offer alternative representations of return distributions. Our contribution concerns the object being predicted and its Bellman recursion. R-C51 keeps the categorical representation but directly predicts a distribution over relative returns.

Modified Bellman operators and uniformization. Bellemare et al. (2016) develop optimality-preserving operators that increase action gaps. Our transformation instead preserves the original action-value differences in expectation for each fixed policy. Transition-dependent discounts make termination explicit (White, 2017), but unequal discounts obstruct a relative backup. Inspired by classical uniformization (Puterman, 2014), we introduce self-loops and rescale rewards to obtain a common continuation discount. This transformation preserves expected values, although it does not need to preserve the full return distributions.

Distributional Bellman theory. Our contraction argument follows the fixed-policy distributional Bellman framework of Bellemare et al. (2017), applied to the constructed process on pairs of state–action pairs. The guarantee concerns the unprojected operator. Categorical approximation requires separate analysis. Rowland et al. (2018) establish convergence properties of categorical distributional reinforcement learning using the Cramér distance. R-C51 adopts the standard categorical projection, but its learned continuation model, function approximation, and changing control policy lie beyond our unprojected fixed-policy guarantee.

B Proofs

B.1 The issue of unequal transition discounts

We show why a critic that learns only return differences cannot identify the target distribution when the two transition discounts differ. This result rules out recovering the sampled target from shift-invariant bootstrap information alone and motivates the value-preserving construction in Section 5.

Theorem B.1. *Fix two sampled transitions (x, r_x, δ_x, x') and (y, r_y, δ_y, y'), and consider the target Y_{xy} in Equation 6. Suppose the critic's bootstrap information is invariant under the common shift*

$$G_u \longmapsto G_u + b$$

for every successor state–action pair u and every constant b. If $\delta_x \neq \delta_y$, then the distribution of Y_{xy} cannot be determined from this information alone. In particular, the shifted returns produce the same critic representation but the target becomes

$$Y_{xy}^{(b)} = Y_{xy} + b(\delta_x - \delta_y).$$

Proof. Under a common shift, every return difference is unchanged

$$(G_u + b) - (G_v + b) = G_u - G_v.$$

The relative critic therefore receives the same bootstrap information. Direct substitution into Equation 6 gives

$$\begin{aligned} Y_{xy}^{(b)} &= r_x - r_y + \delta_x(G_{x'} + b) - \delta_y(G_{y'} + b) \\ &= Y_{xy} + b(\delta_x - \delta_y). \end{aligned}$$

When $\delta_x \neq \delta_y$, any nonzero b changes the target by a nonzero constant while leaving the critic representation fixed. □

B.2 Value preservation and contraction

We first show that uniformization changes the form of the Bellman update without changing the value it represents. This is essential because the relative method should keep the original policy values and optimal actions. We also show that the common continuation coefficient gives the uniformized operator the same contraction factor $\bar{\gamma}$ at every state–action pair.

Proposition B.2 (Value Preservation)**.** *Let $(\bar{T}^\pi q)(x) := \mathbb{E}[\bar{R}_x + \bar{\gamma} q(\bar{X}') \mid x]$, where the expectation includes the environment transition and the constructed successor. For every bounded value function q,*

$$(\bar{T}^\pi q)(x) = q(x) + \rho_x\big((T^\pi q)(x) - q(x)\big). \tag{33}$$

Consequently, for bounded expected rewards, $\bar{T}^\pi$ and T^π have the same unique fixed point for every policy, and the corresponding greedy operators share $Q^\star$. Moreover, $\bar{T}^\pi$ is a $\bar{\gamma}$-contraction in the supremum norm.

Proof of Proposition B.2. Averaging the uniformized target first over the constructed successor and then over the environment transition gives

$$\begin{aligned}(\bar{T}^\pi q)(x) &= \rho_x (T^\pi q)(x) + (\bar{\gamma} - \rho_x c_x) q(x) \\ &= \rho_x (T^\pi q)(x) + (1 - \rho_x) q(x) \\ &= q(x) + \rho_x \big((T^\pi q)(x) - q(x) \big).\end{aligned}$$

The second equality uses $\rho_x(1 - c_x) = 1 - \bar{\gamma}$. This proves Equation 33. Since $\rho_x > 0$,

$$(\bar{T}^\pi q)(x) = q(x) \quad \Longleftrightarrow \quad (T^\pi q)(x) = q(x).$$

The two Bellman equations therefore have the same fixed point. The same identity applied to the greedy operators shows that they share $Q^\star$.

For the contraction claim, let q and v be two bounded functions. By Equation 33, the reward terms immediately cancel, so at every state–action pair,

$$(\bar{T}^\pi q)(x) - (\bar{T}^\pi v)(x) = (1 - \rho_x)(q(x) - v(x)) + \rho_x \mathbb{E}[\Gamma_x (q(X') - v(X')) \mid x].$$

Since $0 < \rho_x \leq 1$ and $\Gamma_x \geq 0$, the triangle inequality gives

$$\begin{aligned}\big| (\bar{T}^\pi q)(x) - (\bar{T}^\pi v)(x) \big| &\leq (1 - \rho_x) |q(x) - v(x)| + \rho_x \mathbb{E}[\Gamma_x |q(X') - v(X')| \mid x] \\ &\leq (1 - \rho_x) \|q - v\|_\infty + \rho_x \mathbb{E}[\Gamma_x \mid x] \|q - v\|_\infty \\ &= \big[(1 - \rho_x) + \rho_x c_x \big] \, \|q - v\|_\infty \\ &= \big[1 - \rho_x (1 - c_x) \big] \|q - v\|_\infty \\ &= \bar{\gamma} \|q - v\|_\infty.\end{aligned}$$

Equation 13 gives the last equality. Taking the supremum over x proves that $\bar{T}^\pi$ is a $\bar{\gamma}$-contraction. □

B.3 The relative distributional operator

The common discount makes repeated fixed-policy updates converge to a unique collection of return-difference distributions with the correct means.

Theorem B.3 (Correct mean and contraction)**.** *Fix a policy π and $p \in [1, \infty]$. Assume uniformly bounded p-th reward moments (uniformly bounded support for $p = \infty$). Each unprojected update $\mathcal{U}^\pi$ in Equation 23 shrinks the largest p-Wasserstein distance between two collections of predictions by a factor of at most $\bar{\gamma} < 1$. Starting from any collection with the same moment condition, repeated updates converge to the unique fixed point: the distributions of $\Delta H(x, y)$, whose means satisfy*

$$\mathbb{E}[\Delta H(x, y)] = Q^\pi(x) - Q^\pi(y).$$

Proof. We adapt the fixed-policy Wasserstein contraction argument of Bellemare et al. (2017, Lemma 3) to the uniformized process on pairs of state–action pairs. Its reward is $\bar{R}_x - \bar{R}_y$, its successor is $(\bar{X}', \bar{Y}')$, and its discount is $\bar{\gamma}$. The two branches are sampled independently, as required by the definition of ΔH in Equation 22.

Compare two collections of predictions μ and ν, using the same joint sample of rewards and successor pairs in both updates. Conditional on this sample, couple the two future predictions optimally, so that the conditional L^p norm of their difference equals the corresponding p-Wasserstein distance (using the essential supremum for $p = \infty$). The reward terms then cancel, and the remaining future differences are multiplied by $\bar{\gamma}$. No independence between the reward difference and the successor pair is needed. Taking the supremum over pairs gives

$$\bar{d}_p(\mathcal{U}^\pi \mu, \mathcal{U}^\pi \nu) \leq \bar{\gamma} \, \bar{d}_p(\mu, \nu), \qquad \bar{d}_p(\mu, \nu) := \sup_{x,y} W_p(\mu(x, y), \nu(x, y)).$$

Since $\bar{\gamma} < 1$, Banach's fixed-point theorem guarantees a unique fixed point and convergence to it on the stated complete space. The distributions of ΔH already satisfy Equation 23, so they are this fixed point. Finally, the expected uniformized return is a fixed point of $\bar{T}^\pi$, so Proposition B.2 gives $\mathbb{E}[H_x] = Q^\pi(x)$. □

Table 2: **Atari Training Configuration**: R-C51

Component	Setting
Discount and support	$\bar{\gamma} = \gamma = 0.99$, 51 atoms between $[-10, 10]$
Comparison and control	Ordered concatenation of e_x, e_y. Action score from Equation 31
Continuation	BCE on $1 - d$ with loss weight 0.33. Gradients through the encoder.
Optimizer	Adam (Kingma & Ba, 2014) with $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 0.01/32$
Learning rate	3×10^{-4}
Updates	Batch size 32. Learning starts at 80,000 transitions, one update every 3 environment transitions
Target network	Selects and evaluates successor actions, hard copy every 10,000 environment steps
Replay	Capacity 10^6, uniform sampling
Environment	Two EnvPool actors, frame skip 4, up to 30 initial no-ops, fire reset, reward clipping and episodic life enabled
Exploration	ϵ decreases linearly from 1 to 0.01 over the first million transitions
Evaluation	100 full games with $\epsilon = 0.001$, reporting average return

C Implementation and Training Details

This section specifies the update and Atari training setup for R-C51. We use the target mixture and action score defined in Section 6, with $\bar{\gamma} = \gamma$. Table 2 summarizes the training configuration (Machado et al., 2018) and Algorithm 1 shows the general update steps.

C.1 Training Losses and Minibatch Update

Each update samples B transitions independently from the replay buffer and builds pairs from it. Let $q_\theta(u)$ denote the online continuation head's predicted probability of nontermination at the state–action pair u. We minimize

$$\mathcal{L} = \frac{1}{B}\sum_{i=1}^{B} \mathcal{L}_{\mathrm{RC51}}(x_i, y_i) + \beta_{\mathrm{cont}} \frac{1}{B}\sum_{j=1}^{B} \mathrm{BCE}\big(q_\theta(u_j), 1 - d_j\big), \qquad \beta_{\mathrm{cont}} = 0.33, \tag{34}$$

where u_j ranges over all sampled state–action pairs and d_j is the observed termination indicator. A life loss is also treated as termination. Both losses update the shared encoder and action-conditioned representation. We use θ for all online parameters and θ^- for their target model.

C.2 Network Architecture

For Atari, we use the same CNN encoder as C51 (Bellemare et al., 2017). The network first encodes each state–action pair independently. The encoder maps the state observation to a feature vector. We concatenate these features with a learned action embedding (48-dim) and pass the result through a two-layer MLP (512 units) to obtain a representation e_x of $x = (s, a)$.

A two-layer MLP (512 units) takes the ordered concatenation $[e_x, e_y]$ and produces N logits. Applying a softmax gives the categorical probabilities $p_\theta(x, y)$ on $\mathcal{Z}$. This head predicts the return-difference distribution directly. Because reversing the inputs is not constrained to reflect the predicted distribution, the two comparison directions can yield different mean estimates, motivating their combination in Section 6.2.

A separate linear head takes e_x and applies a sigmoid to predict the probability of nontermination. Multiplying this probability by γ gives $\hat{c}_x$, the expected transition-discount estimate used in the target construction. Both the distributional loss and the continuation loss update the shared representation.

C.3 Computational Cost.

Under the same EnvPool setup, an R-C51 run takes approximately 4.5 hours, compared with 3 hours for C51, corresponding to roughly 50% longer training time. These experiments use an H200 MIG

1g.18gb instance with eight CPU cores and 8 GB RAM. The performance improvements therefore come with additional computational cost.

Algorithm 1 One update of R-C51

Require: online parameters θ, target parameters θ^-, replay $\mathcal{B}$, batch size B

1: Sample B transitions from $\mathcal{B}$ and form B pairs (always different transitions as pair).
2: **for** each transition pair (τ_x, τ_y), in parallel **do**
3: Using the target network, compute $\widehat{c}_u$, ρ_u, $\bar{r}_u$, and g_u for $u \in \{x, y\}$.
4: Select successor actions with S_{θ^-} (Equation 31); set $u' = u$ if $d_u = 1$.
5: Form the four-way mixture ν (Equation 27).
6: Project atoms $\bar{r}_x - \bar{r}_y + \gamma\zeta_k$ with masses ν_k to obtain the target m.
7: **end for**
8: Take one Adam step on Equation 34.
9: Copy θ to θ^- when the update interval is reached.

D SINGLE GAME PERFORMANCE

Table 3 presents the performance of each algorithm on individual games.

Table 3: **Atari Evaluation Returns**. Mean and ± sample standard deviation for each individual game. C51, R-C51, and C51-U use 5 seeds, while C51-MLP, and C51-UPS use 3. Bold entries have the highest mean in each row.

Game	C51	R-C51	C51-U	C51-MLP	C51-UPS
Alien	1671.6 ± 232.8	$\mathbf{2381.2 \pm 257.7}$	1614.7 ± 91.6	1964.7 ± 182.9	1669.2 ± 41.6
Amidar	505.7 ± 135.8	$\mathbf{836.6 \pm 99.7}$	611.1 ± 93.4	526.5 ± 68.8	437.9 ± 80.4
Assault	4208.1 ± 1085.4	4157.9 ± 332.7	$\mathbf{4811.3 \pm 691.0}$	4078.2 ± 1756.2	4261.2 ± 2112.4
Asterix	35637.2 ± 9010.9	22557.6 ± 3540.8	27172.6 ± 3079.0	21359.1 ± 3680.6	$\mathbf{69622.6 \pm 21329.7}$
Asteroids	869.6 ± 379.8	$\mathbf{1588.8 \pm 160.5}$	998.8 ± 292.7	1140.6 ± 448.7	559.9 ± 175.7
Atlantis	$\mathbf{705283.3 \pm 84018.5}$	555422.8 ± 54553.0	686332.8 ± 77639.2	9059.3 ± 2144.4	526055.6 ± 324028.1
BankHeist	1230.8 ± 50.9	1120.0 ± 149.8	1098.2 ± 97.1	$\mathbf{1245.0 \pm 208.0}$	681.5 ± 409.0
BattleZone	22268.8 ± 8774.6	22302.0 ± 1258.7	19386.0 ± 5225.9	$\mathbf{28130.0 \pm 1556.9}$	25250.0 ± 2279.2
BeamRider	6709.7 ± 496.1	8143.7 ± 564.9	9669.2 ± 703.2	7687.8 ± 436.2	$\mathbf{11719.3 \pm 2932.7}$
Berzerk	631.4 ± 138.8	806.3 ± 119.7	838.4 ± 247.5	809.7 ± 128.1	$\mathbf{867.1 \pm 338.9}$
Bowling	32.3 ± 12.8	32.4 ± 5.1	32.2 ± 14.2	$\mathbf{32.8 \pm 5.2}$	32.4 ± 16.8
Boxing	83.9 ± 3.9	$\mathbf{92.7 \pm 1.6}$	84.8 ± 6.1	86.6 ± 3.4	83.2 ± 9.0
Breakout	$\mathbf{422.0 \pm 39.9}$	348.0 ± 18.4	359.2 ± 34.4	380.6 ± 31.1	406.7 ± 30.2
Centipede	6745.1 ± 696.3	1538.2 ± 116.4	6386.8 ± 395.0	6530.2 ± 490.7	$\mathbf{6912.1 \pm 216.9}$
ChopperCommand	1183.4 ± 192.6	3698.4 ± 253.1	2916.0 ± 523.3	$\mathbf{3747.0 \pm 2538.9}$	3391.0 ± 1029.4
CrazyClimber	143477.8 ± 10654.2	130596.4 ± 5099.2	142010.7 ± 5919.3	132366.3 ± 7283.6	$\mathbf{143677.6 \pm 4508.9}$
Defender	4641.8 ± 1129.7	12615.1 ± 1581.3	10260.8 ± 3968.3	15272.1 ± 10371.9	$\mathbf{16637.5 \pm 2376.4}$
DemonAttack	$\mathbf{59712.6 \pm 18178.9}$	48116.8 ± 11693.8	55791.2 ± 28522.0	52951.3 ± 30103.5	36774.8 ± 17394.1
DoubleDunk	-13.7 ± 3.9	-13.1 ± 1.8	-13.8 ± 1.7	-15.3 ± 5.1	$\mathbf{-10.1 \pm 3.1}$
Enduro	1878.4 ± 197.3	$\mathbf{1915.0 \pm 90.8}$	1771.6 ± 68.2	1461.8 ± 736.2	1658.9 ± 326.8
FishingDerby	-5.4 ± 10.0	-4.9 ± 9.7	$\mathbf{7.0 \pm 9.4}$	-10.8 ± 13.7	-1.9 ± 3.1
Freeway	33.5 ± 0.3	33.7 ± 0.4	33.7 ± 0.1	33.2 ± 0.1	$\mathbf{33.8 \pm 0.1}$
Frostbite	4017.2 ± 929.3	$\mathbf{4519.0 \pm 863.0}$	2646.5 ± 927.8	3089.1 ± 1002.4	4058.3 ± 638.2
Gopher	9880.8 ± 1590.3	9912.0 ± 3170.3	$\mathbf{10314.9 \pm 1812.4}$	7248.2 ± 493.9	9886.4 ± 1268.4
Gravitar	212.9 ± 65.3	227.9 ± 54.9	147.8 ± 57.4	$\mathbf{297.1 \pm 112.2}$	206.1 ± 69.0
Hero	16253.9 ± 2250.0	20544.5 ± 756.6	$\mathbf{20730.9 \pm 2392.0}$	17983.2 ± 2907.3	19936.9 ± 2991.9
IceHockey	$\mathbf{-6.0 \pm 1.1}$	-6.5 ± 0.6	-9.4 ± 2.6	-9.5 ± 3.4	-9.7 ± 2.7
Jamesbond	620.6 ± 131.1	$\mathbf{708.5 \pm 39.7}$	624.3 ± 171.7	577.8 ± 339.4	690.1 ± 116.7
Kangaroo	6859.0 ± 4696.9	4876.2 ± 3291.8	$\mathbf{11431.8 \pm 3960.1}$	7835.0 ± 1574.5	10306.3 ± 4335.6
Krull	$\mathbf{9270.6 \pm 317.6}$	7070.5 ± 1508.6	8342.8 ± 602.7	9173.3 ± 359.4	7899.6 ± 1419.5
KungFuMaster	18979.5 ± 2474.8	$\mathbf{25255.8 \pm 3992.9}$	22419.0 ± 3393.5	20857.0 ± 833.0	20710.3 ± 2663.0
MontezumaRevenge	0.0 ± 0.0	$\mathbf{1.8 \pm 2.6}$	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
MsPacman	2278.8 ± 345.5	$\mathbf{3247.1 \pm 567.5}$	2399.2 ± 140.9	2612.6 ± 151.3	2672.7 ± 61.6
NameThisGame	9065.3 ± 709.8	4215.2 ± 264.1	$\mathbf{9222.3 \pm 647.7}$	7648.1 ± 425.8	8187.3 ± 1478.6
Phoenix	12723.2 ± 5056.2	$\mathbf{15848.6 \pm 1194.6}$	10522.6 ± 2035.4	12739.9 ± 6929.7	11464.8 ± 2250.9
Pitfall	-4.9 ± 6.7	-6.9 ± 6.6	-9.5 ± 9.2	$\mathbf{-0.3 \pm 0.5}$	-10.7 ± 15.2
Pong	20.5 ± 0.4	$\mathbf{20.9 \pm 0.0}$	20.6 ± 0.3	20.6 ± 0.2	20.1 ± 0.4
PrivateEye	2276.0 ± 4860.1	$\mathbf{5049.0 \pm 6944.8}$	83.2 ± 19.0	87.3 ± 171.7	103.1 ± 78.8
Qbert	15034.3 ± 1404.3	$\mathbf{18643.5 \pm 1564.4}$	15773.4 ± 986.4	15618.6 ± 1638.8	16567.0 ± 1724.5
Riverraid	12367.9 ± 2151.7	12548.1 ± 331.6	13635.2 ± 1057.6	13623.4 ± 1168.3	$\mathbf{14180.7 \pm 425.5}$
RoadRunner	53317.0 ± 2825.6	56815.6 ± 2023.9	58732.0 ± 3623.8	51481.6 ± 3421.0	$\mathbf{59678.5 \pm 2477.5}$
Robotank	32.2 ± 1.1	$\mathbf{48.9 \pm 6.4}$	32.8 ± 4.1	30.1 ± 6.1	31.0 ± 2.3
Seaquest	8529.6 ± 3095.0	11324.6 ± 4038.2	7012.7 ± 1485.6	7054.7 ± 656.6	$\mathbf{19938.6 \pm 11580.5}$
Skiing	-25249.5 ± 4993.2	-25946.5 ± 5092.2	-23161.0 ± 4132.7	-29973.2 ± 0.6	$\mathbf{-21310.7 \pm 2856.9}$
Solaris	221.3 ± 260.3	$\mathbf{1750.5 \pm 271.7}$	1488.6 ± 796.8	391.2 ± 549.3	400.2 ± 475.9
SpaceInvaders	$\mathbf{2772.6 \pm 1119.1}$	2088.3 ± 193.4	2088.3 ± 329.7	1892.3 ± 74.2	2378.6 ± 69.9
StarGunner	9300.2 ± 4208.2	$\mathbf{60347.0 \pm 1712.3}$	47352.6 ± 4240.1	34162.0 ± 8265.0	13933.6 ± 14719.4
Surround	$\mathbf{4.3 \pm 1.8}$	-3.6 ± 2.9	-4.0 ± 3.6	1.9 ± 1.0	-0.5 ± 6.3
Tennis	-7.8 ± 10.1	$\mathbf{-0.9 \pm 0.0}$	-12.0 ± 7.4	-1.3 ± 0.8	-8.9 ± 9.3
TimePilot	4079.0 ± 1156.9	$\mathbf{7358.4 \pm 498.8}$	4169.8 ± 567.4	3270.6 ± 1314.2	4723.0 ± 444.1
Tutankham	182.5 ± 46.5	152.1 ± 44.7	$\mathbf{221.6 \pm 10.9}$	201.2 ± 35.1	200.5 ± 10.7
UpNDown	8398.2 ± 2740.0	8192.0 ± 599.4	3201.3 ± 1033.1	$\mathbf{10015.3 \pm 2418.1}$	5516.5 ± 1394.7
Venture	0.0 ± 0.0	$\mathbf{306.2 \pm 422.5}$	147.0 ± 328.7	0.0 ± 0.0	155.3 ± 269.0
VideoPinball	310876.0 ± 31238.7	366061.3 ± 94385.7	458331.2 ± 31211.2	384479.0 ± 83484.2	$\mathbf{465488.8 \pm 40145.2}$
WizardOfWor	$\mathbf{3356.1 \pm 950.0}$	3131.8 ± 664.1	3179.8 ± 867.0	2778.3 ± 549.1	2025.6 ± 343.5
YarsRevenge	11583.2 ± 2029.7	11926.9 ± 1340.6	10137.4 ± 1281.5	$\mathbf{13835.3 \pm 5630.1}$	10010.4 ± 284.4
Zaxxon	4614.1 ± 1261.6	$\mathbf{10037.4 \pm 393.3}$	7374.2 ± 1335.4	7296.6 ± 1731.2	6441.3 ± 1471.6