RELATIVE VALUE LEARNING

Marc Höftmann, Jan Robine & Stefan Harmeling
Department of Computer Science, Technical University of Dortmund, Germany
Lamarr Institute for Machine Learning and Artificial Intelligence
{marc.hoeftmann,jan.robine,stefan.harmeling}@tu-dortmund.de

ABSTRACT

In reinforcement learning, critics typically estimate absolute state values $V(s)$, estimating how good a particular situation is in isolation. However, it turns out that only differences in value are relevant for control. Motivated by this, we propose *Relative Value Learning* (RV), a framework that learns value differences directly via an antisymmetric function $\Delta(s_i, s_j) = V(s_i) - V(s_j)$. We introduce a pairwise Bellman operator and prove it is a γ-contraction with a unique fixed point equal to the true value differences, derive well-posed 1-step, n-step and λ-return targets and reconstruct generalized advantage estimation from pairwise differences to obtain an unbiased policy-gradient estimator (R-GAE). Beyond theoretical results, we integrate RV with PPO and achieve competitive performance on the Atari benchmark (49 ALE games) compared to standard PPO, indicating that relative value estimation is an effective alternative to absolute critics. Our code is available at https://github.com/Hauf3n/relative-value-learning.

1 MOTIVATION

In control, actions are chosen by *comparisons*, not by absolute magnitudes. What matters is how good one state (or action) is *relative* to another. Formally, for $\gamma < 1$, V^π is uniquely determined by the Bellman equation, but shifting it by any constant leaves advantages and greedy choices unchanged. This gauge freedom makes the absolute scale behaviorally meaningless: only differences matter. Advantages $A^\pi(s, a)$ or greedy action selection $\max_a Q^\pi(s, a)$ imply that absolute scales are not behaviorally meaningful. *Optimal control depends not on absolute magnitudes but exclusively on relative differences.* Despite this, standard value-based RL trains a critic to approximate V^π (or Q^π) and treats differences as a derived quantity. This introduces unnecessary degrees of freedom (the unpinned offset), invites drift under reward shaping or baseline changes, and can be ill-posed in settings where only comparisons or implicit feedback are given (e.g., preference-based or human-in-the-loop RL), precisely where the absolute scale is ambiguous while pairwise relations remain well-defined. A formulation that places *relative* information at the center would match the invariances of the decision making problem itself.

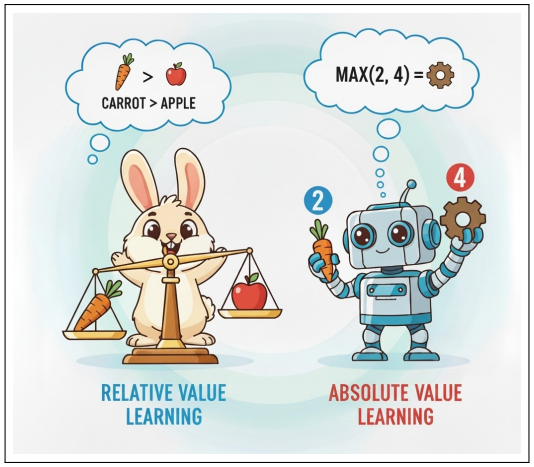

Figure 1: **Relative Value vs. Absolute Value Learning.**. RV (left) learns value differences between states for decision making while AV (right) learns the value for each state in isolation and then decides for the best decision (e.g. by taking maximum in Q-learning).

We therefore adopt the following viewpoint: make value differences the *primary* learning objective. Concretely, we learn an antisymmetric function $\Delta_\theta : S \times S \to \mathbb{R}$ with $\Delta_\theta(s_i, s_j) = -\Delta_\theta(s_j, s_i)$ that approximates $V^\pi(s_i) - V^\pi(s_j)$. Working directly with Δ_θ eliminates the gauge degree of freedom by construction and aligns the critic with the invariants already exploited by policy-gradient methods

through baselines. In addition, advantages can be reconstructed from pairwise differences without knowing the absolute value of any state, providing an unbiased policy-gradient estimator (R-GAE). This perspective is not just aesthetic. It enables a clean analytic foundation.

Our Contributions:

1. **Pairwise Value Operator.** We formalize a Bellman operator on antisymmetric functions and prove γ-contraction with a fixed point equal to true value differences.
2. **Value Targets.** We derive 1-step / n-step / λ-return targets using only observable rewards and non terminal pairwise terms, ensuring well-posed bootstrapping targets.
3. **Relative GAE (R-GAE).** We show that GAE can be reconstructed from pairwise differences that also results in an *unbiased* policy gradient estimator. In addition we derive the relationship between GAE and R-GAE and show that RV achieves competitive performance on Atari.

2 Related Work

Classical value-based RL trains absolute state or action values using Bellman operators, e.g. TD(λ), DQN, Double DQN, Rainbow, and other actor critic variants (Sutton & Barto, 2018; Mnih et al., 2013; Van Hasselt et al., 2016; Wang et al., 2016a; Hessel et al., 2018) Distributional critics (Bellemare et al., 2017; Dabney et al., 2018) restructure the target but continue to operate in an absolute space. Although adding a constant to V^π or Q^π leaves action preferences unchanged (Sutton & Barto, 2018), absolute critics still predict a scalar on an arbitrary scale. By contrast, RV removes the offset degree of freedom at the model level by learning antisymmetric value differences over state pairs, aligning the function class with the invariances of decision making. This invariance is a special case of policy-invariant reward transformations (potential-based shaping) (Ng et al., 1999). Several methods estimate or emphasize advantages rather than values. Direct Advantage Estimation (DAE) Pan et al. (2022) directly learns advantages $A^\pi(s, a)$ to bypass value learning. Dueling networks decompose learning $Q^\pi(s, a) = V^\pi(s) + A^\pi(s, a)$ to improve sample efficiency and robustness Wang et al. (2016b). Actor-critic variants such as A2C/A3C Mnih et al. (2016), TRPO Schulman et al. (2015a) and clipped PPO Schulman et al. (2017) then use baselines and advantage estimates within trust-region-style updates. In contrast, RV directly learns value differences $\Delta(s_i, s_j)$ with an explicit pairwise Bellman operator to provide the actor *relative advantages*. RV's critic uses a siamese difference head that enforces antisymmetry and zero self-difference by design. Earlier work in stochastic control has also advocated learning value differences rather than absolute values. Bertsekas (1997) introduces differential training of rollout policies, approximating cost-to-go differences with TD-style methods. In RL, pairwise objectives have appeared mainly in preference-based and human-in-the-loop RL (Christiano et al., 2017; Leike et al., 2018) or in inverse RL, but not as a Bellman-consistent value critic. Our work fills this gap.

3 Relative Value Learning

We now present *Relative Value Learning* (RV), which learns an *antisymmetric* function over all state pairs with a neural network

$$\Delta_\theta : \mathcal{S} \times \mathcal{S} \to \mathbb{R}, \qquad \Delta_\theta(s_i, s_j) = -\Delta_\theta(s_j, s_i), \tag{1}$$

that tries to approximate the *value difference* $\Delta^\pi(s_i, s_j) := V^\pi(s_i) - V^\pi(s_j)$, under a fixed policy π. As a result, RV avoids exact value estimates during training and integrates naturally with on-policy actor-critic methods (e.g., PPO) by supplying *relative advantages* (R-GAE).

3.1 Preliminaries

We consider a discounted Markov decision process (MDP) (S, A, P, r, γ) with state space S, action space A, transition function $P(s' \mid s, a)$, bounded reward function $r : S \times A \to [r_{\min}, r_{\max}]$, and discount factor $\gamma \in [0, 1)$. Considering a stochastic policy $\pi(a \mid s)$, the value function under π is

$$V^\pi(s) := \mathbb{E}_{\pi,P}\Big[\sum_{t=0}^{\infty} \gamma^t\, r(s_t, a_t) \,\Big|\, s_0 = s\Big], \tag{2}$$

which satisfies the Bellman equation

$$V^\pi(s) \;=\; r_\pi(s) + \gamma\, \mathbb{E}_{s' \sim P^\pi(\cdot|s)}\big[V^\pi(s')\big], \quad r_\pi(s) := \mathbb{E}_{a\sim\pi(\cdot|s)}[r(s,a)]. \tag{3}$$

Gauge Freedom and Relative Values. Shifted value functions do also satisfy a Bellman equation like Equation 3 (Lemma B.1 explains how to shape the reward). These shifts are irrelevant for control, since the advantages or greedy action selection are invariant. Hence the absolute scale of V^π is not meaningful, and only differences of values are invariant. We therefore define the *pairwise value difference*

$$\Delta^\pi(s_i, s_j) \;=\; V^\pi(s_i) - V^\pi(s_j), \quad (s_i, s_j) \in S \times S. \tag{4}$$

Note that it is antisymmetric $\Delta^\pi(s_i, s_j) = -\Delta^\pi(s_j, s_i)$, and satisfies $\Delta^\pi(s, s) = 0$ for all s.

Pairwise Bellman Identity. Fix $(s_i, s_j) \in S \times S$. Draw $s_i' \sim P^\pi(\cdot \mid s_i)$ and $s_j' \sim P^\pi(\cdot \mid s_j)$ *independently* (only conditional on s_i or s_j respectively). Subtracting the Bellman equations of (3) for s_i and s_j gives the recursive identity

$$\Delta^\pi(s_i, s_j) \;=\; r_\pi(s_i) - r_\pi(s_j) \;+\; \gamma\, \mathbb{E}_{\substack{s_i'\sim P^\pi(\cdot|s_i)\\ s_j'\sim P^\pi(\cdot|s_j)}}\big[\Delta^\pi(s_i', s_j')\big]. \tag{5}$$

Equation 5 depends only on observable one-step rewards via r_π and on pairwise differences at successors. It is invariant to any additive shift of V^π as we have shown in Lemma B.1.

3.2 Pairwise Bellman Operator

Fix a policy π. We work on the Banach space of bounded antisymmetric pairwise functions

$$\mathcal{F} \;:=\; \Big\{\Delta : \mathcal{S}\times\mathcal{S} \to \mathbb{R} \;\Big|\; \Delta(s_i, s_j) = -\Delta(s_j, s_i),\; \|\Delta\|_\infty < \infty\Big\}, \tag{6}$$
$$\|\Delta\|_\infty \;=\; \sup_{(s_i, s_j)} |\Delta(s_i, s_j)|.$$

In addition, let $\Delta r^\pi(s_i, s_j) := r^\pi(s_i) - r^\pi(s_j)$ be the immediate reward difference.

Operator form. Let $\widehat{\mathcal{P}}_\pi$ act on $\mathcal{F}$ by

$$(\widehat{\mathcal{P}}_\pi \Delta)(s_i, s_j) \;=\; \mathbb{E}_{s_i'\sim P^\pi(\cdot|s_i),\, s_j'\sim P^\pi(\cdot|s_j)}\big[\Delta(s_i', s_j')\big]. \tag{7}$$

Define the *pairwise Bellman operator* $T_\pi : \mathcal{F} \to \mathcal{F}$ by

$$(T_\pi \Delta)(s_i, s_j) \;:=\; \Delta r^\pi(s_i, s_j) \;+\; \gamma\, (\widehat{\mathcal{P}}_\pi \Delta)(s_i, s_j). \tag{8}$$

This coincides with the definition from Equation (5) used in our method when s_i' and s_j' are drawn independently from $P^\pi(\cdot|s_i)$ and $P^\pi(\cdot|s_j)$, respectively.

Theorem 3.1 (Contraction and uniqueness)**.** *For any $\Delta_1, \Delta_2 \in \mathcal{F}$,*

$$\|T_\pi \Delta_1 - T_\pi \Delta_2\|_\infty \;\le\; \gamma\, \|\Delta_1 - \Delta_2\|_\infty.$$

Consequently, by the Banach fixed-point theorem, T_π has a unique fixed point $\Delta^\pi \in \mathcal{F}$. Moreover, this fixed point equals the true value differences, i.e., $\Delta^\pi(s_i, s_j) = V^\pi(s_i) - V^\pi(s_j)$ for all $(s_i, s_j) \in \mathcal{S}\times\mathcal{S}$.

Proof. The immediate difference Δr^π cancels in $T_\pi\Delta_1 - T_\pi\Delta_2$, hence for each (s_i, s_j),

$$\begin{aligned}\big|(T_\pi\Delta_1 - T_\pi\Delta_2)(s_i, s_j)\big| \;&=\; \gamma\,\Big|\mathbb{E}\big[(\Delta_1 - \Delta_2)(s_i', s_j')\big]\Big| \;\le\; \gamma\,\mathbb{E}\big[|(\Delta_1 - \Delta_2)(s_i', s_j')|\big] \\ &\le\; \gamma\,\|\Delta_1 - \Delta_2\|_\infty.\end{aligned}$$

Since this bound holds for all pairs (s_i, s_j) thus taking the supremum over (s_i, s_j) concludes with $\|\mathcal{T}\Delta_1 - \mathcal{T}\Delta_2\|_\infty \le \gamma\|\Delta_1 - \Delta_2\|_\infty$. □

Related 'relative' value functions arise in average-reward MDPs, where values are defined only up to an additive constant and algorithms fix the gauge via relative value iteration (Abounadi et al., 2001; Puterman, 2014; Bertsekas, 2025).

3.3 RELATIVE GENERALIZED ADVANTAGE ESTIMATION (R-GAE)

In this section, we define the R-GAE estimator which is in essence analogous to GAE (Schulman et al., 2015b). Furthermore, we derive the relationship between GAE and R-GAE and show that both estimators learn the same (optimal) policy. Then, R-GAE is combined with PPO (Schulman et al., 2017) to provide *relative advantages*. Hereby, one of our key theoretical contributions is the fact that GAE can be reconstructed without knowing the exact value of a state.

Relative GAE. First, we construct *relative values* $\tilde{V}_\theta$ for each state of an arbitrary environment rollout $(s_0, s_1, \ldots, s_T)$, where T denotes the rollout length. Note in our notation that s_0 is *not* the environment's start state, but it can be any state. This notation simplifies the following descriptions. We define the relative value sequence by telescoping the differences:

$$\tilde{V}_\theta(s_0) := 0, \quad \tilde{V}_\theta(s_t) := \sum_{k=0}^{t-1} \Delta_\theta(s_{k+1}, s_k) \quad (t \geq 1). \tag{9}$$

Note, that it is also possible to use a larger step size for calculating relative values, e.g. one can directly compute $\tilde{V}_\theta(s_t) = \Delta_\theta(s_t, s_0)$. If $\Delta_\theta = \Delta^\pi$, then the relationship becomes $\tilde{V}_\theta(s_t) = V^\pi(s_t) - V^\pi(s_0)$. Analogous to GAE, define relative TD residuals

$$\tilde{\delta}_t \;:=\; r_t + \gamma \tilde{V}_\theta(s_{t+1}) - \tilde{V}_\theta(s_t), \tag{10}$$

and finally construct the relative GAE similarly

$$\tilde{A}_t \;:=\; \sum_{l=0}^{T-t} (\gamma\lambda)^l \tilde{\delta}_{t+l}. \tag{11}$$

Essentially, RV acts as the critic and replaces GAE with R-GAE in the PPO clipping objective. Furthermore, we give additional theoretical insight about R-GAE in the following. In short, these insights conclude that:

1. There exists a relationship between GAE and R-GAE that is given as $\tilde{A}_t \;=\; A_t + B_t$. For more details, see Lemma 3.2.

2. The optimal policy for both estimators is identical. See Corollary 3.3.

Lemma 3.2 (Relationship between GAE and R-GAE)**.** *We call $C := V^\pi(s_0)$ the trajectory constant. If $\Delta_\theta = \Delta^\pi$, then the following equality holds*

$$\tilde{A}_t \;=\; A_t + B_t, \tag{12}$$

where A_t is the standard GAE computed from V^π, and $B_t \;=\; (1-\gamma)C \sum_{l=0}^{T-t} (\gamma\lambda)^l$.

Proof. By construction of the relative values, we have that $\tilde{V}(s_t) = V^\pi(s_t) - C$ for all t. Hence inserting this form in Equation 10 gives

$$\tilde{\delta}_t = r_t + \gamma\big(V^\pi(s_{t+1}) - C\big) - \big(V^\pi(s_t) - C\big) = \underbrace{r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)}_{\delta_t} + (1-\gamma)\, C, \tag{13}$$

where δ_t is the TD residual. Therefore, for each timestep the relative advantage is

$$\tilde{A}_t = \sum_{l=0}^{T-t} (\gamma\lambda)^l\, \tilde{\delta}_{t+l} = \sum_{l=0}^{T-t} (\gamma\lambda)^l\, \delta_{t+l} \;+\; (1-\gamma)\, C \sum_{l=0}^{T-t} (\gamma\lambda)^l = A_t + B_t,$$

□

Gauge Invariance vs. Trajectory-constant Baseline. To get a better understanding of the trajectory constant B_t, we discuss its role and properties in Appendix C. Lemma B.1 states that adding any scalar $c \in R$ to V^π (together with the corresponding reward shaping) leaves action preferences, advantages and pairwise value differences invariant; hence the absolute gauge of V^π is behaviorally irrelevant. Lemma 3.2, however, shows when we reconstruct advantages from pairwise differences by

anchoring to zero on each rollout, there exists indeed a difference quantified by B_t. These statements are compatible and are not contradicting each other. Anchoring $\tilde{V}(s_0) = 0$ is a gauge fixing analogous to the differential (relative) value normalization used in average-reward MDPs (Abounadi et al., 2001; Puterman, 2014; Bertsekas, 2025). The only difference is scope: Ours is per-trajectory and theirs is global, so unbiasedness (see Corollary 3.3) is preserved while a trajectory-constant B_t may appear.

As a remark, we can achieve pointwise equality with GAE, if we learned the *non-antisymmetric* two-argument function

$$\Delta_\gamma(s', s) := \gamma V(s') - V(s), \tag{14}$$

so that the temporal-difference residual $\delta_t = r_t + \Delta_\gamma(s_{t+1}, s_t)$ is modeled precisely. The corresponding Bellman operator is indeed a contractive self-map on the pairwise space, which can be derived analogous to Theorem 3.1. However, we tried to apply $\Delta_\gamma(s', s)$ in practice, but the results are worse compared to R-GAE. Therefore, we continue to work with R-GAE and show in Section 4 how to reduce the variance of B_t.

Corollary 3.3 (Unbiasedness for policy gradient)**.** *Let $\nabla_\phi J(\phi) := \mathbb{E}_t[\nabla_\phi \log \pi_\phi(a_t \mid s_t)\, A_t]$ denote the standard policy gradient (PG) under π_ϕ. If the advantages in the score-function estimator are replaced by the relative advantages from Lemma 3.2, i.e.,*

$$\nabla_\phi \tilde{J}(\phi) \;:=\; \mathbb{E}_t\Big[\nabla_\phi \log \pi_\phi(a_t \mid s_t)\, \tilde{A}_t\Big],$$

then

$$\nabla_\phi \tilde{J}(\phi) \;=\; \nabla_\phi J(\phi). \tag{15}$$

Proof. By Lemma 3.2, $\tilde{A}_t = A_t + B_t$ with $B_t = \frac{C\left(1-(\gamma\lambda)^{T-t+1}\right)}{1-\gamma\lambda}$ as trajectory constant, hence

$$\mathbb{E}_t\Big[\nabla_\phi \log \pi_\phi(a_t \mid s_t)\, \tilde{A}_t\Big] = \mathbb{E}_t[\nabla_\phi \log \pi_\phi(a_t \mid s_t)\, A_t] + \mathbb{E}_t[\nabla_\phi \log \pi_\phi(a_t \mid s_t)\, B_t]\,. \tag{16}$$

Condition on s_t by the trajectory prefix up to time t (so that B_t does not depend on a_t) and then using the score-function identity gives

$$\mathbb{E}[\nabla_\phi \log \pi_\phi(a_t \mid s_t)\, B_t \mid s_t] = B_t\, \mathbb{E}_{a_t \sim \pi_\phi(\cdot|s_t)}[\nabla_\phi \log \pi_\phi(a_t \mid s_t)] = 0. \tag{17}$$

Taking expectations over t yields

$$\mathbb{E}_t\Big[\nabla_\phi \log \pi_\phi(a_t \mid s_t)\, \tilde{A}_t\Big] = \mathbb{E}_t[\nabla_\phi \log \pi_\phi(a_t \mid s_t)\, A_t]\,, \tag{18}$$

which is precisely Equation 15. □

3.4 Relative Value Targets

When approximating the pairwise Bellman operator (Eq. 8) from two sampled 1-step transitions $\tau_i = (s_i, a_i, r_i, d_i, s_{i+1})$, $\tau_j = (s_j, a_j, r_j, d_j, s_{j+1})$, terminal successor states make the naive bootstrap $\Delta(s_{i+1}, s_{j+1})$ ill-posed because absolute values are not available in our formulation. For example, if s_{i+1} is a terminal state ($d_i = 1$), then we need to calculate

$$\Delta(s_{i+1}, s_{j+1}) = 0 - V(s_{j+1}) = -V(s_{j+1}),$$

which is simply not accessible in that way. Therefore, we need to rearrange all bootstrapping targets in terms of observable rewards and non terminal pairwise differences $\Delta_\theta(\cdot, \cdot)$. In this way the targets get well-posed and are compatible with the operator T_π on antisymmetric functions. Due to limited space, the complete formal derivation is given in Appendix A.

1-step Target. Given the prediction $\Delta_\theta(s_i, s_j)$, the corresponding 1-step target takes the form

$$y_{ij}^{(1)} \;:=\; (r_i - r_j) \;+\; \gamma\, \delta_{ij}, \tag{19}$$

where the bootstrap term δ_{ij} depends on the successor terminal flags $d_i, d_j \in \{0, 1\}$:

$$\delta_{ij} \;=\; \begin{cases} \Delta_\theta(s_{i+1}, s_{j+1}), & \text{if } d_i = 0,\ d_j = 0, \\ \Delta_\theta(s_{i+1}, s_j) \;+\; r_j, & \text{if } d_i = 0,\ d_j = 1, \\ \Delta_\theta(s_i, s_{j+1}) \;-\; r_i, & \text{if } d_i = 1,\ d_j = 0, \\ \Delta_\theta(s_i, s_j) \;+\; r_j \;-\; r_i, & \text{if } d_i = 1,\ d_j = 1. \end{cases} \tag{20}$$

When $d_i{=}d_j{=}1$, one may replace the last line of Equation 20 by $\delta_{ij}{=}0$ (both successors are absorbing with zero value) to reduce variance at ends of the two episodes. We use $\delta_{ij}{=}0$ by default, so the derived fourth case in Appendix A is optional.

N-step Target. To extend beyond 1-step temporal difference targets, we define the pairwise n-step case by considering two trajectories

$$\tau_i = \{(s_{i+k}, r_{i+k}, d_{i+k}, s_{i+k+1})\}_{k\geq 0}, \qquad \tau_j = \{(s_{j+k}, r_{j+k}, d_{j+k}, s_{j+k+1})\}_{k\geq 0}.$$

Then, the target becomes

$$y_{ij}^{(n)} := \sum_{k=0}^{n-1} \gamma^k (r_{i+k} - r_{j+k}) + \gamma^n \Delta_\theta(s_{i+n}, s_{j+n}), \tag{21}$$

with the assumption that neither trajectory terminates within the n-step window (i.e., $d_{i+k} = d_{j+k} = 0$ for $k < n$). If the trajectories have different length, then we take the minimum length. Note, that the final estimated difference $\Delta_\theta(s_{i+n}, s_{j+n})$ needs to consider the case distinction from Eq. 20.

λ-Return. Interpolating between high-bias/low-variance ($n{=}1$) and Monte-Carlo limits gives the pairwise λ-return

$$y_{ij}^{(\lambda)} \;:=\; (1-\lambda) \sum_{n=1}^{\infty} \lambda^{\,n-1}\, y_{ij}^{(n)}, \qquad \lambda \in [0,1], \tag{22}$$

with the convention that $y_{ij}^{(n)}$ is truncated at the first encountered terminal using the case distinction from Equation 20.

4 Relative Value Initialization

As demonstrated in Lemma 3.2, the zero-anchor $\tilde{V}(s_0) = 0$ (see Equation 9) induces a trajectory-constant offset B_t for R-GAE. If the unknown $|C|$ is large (recall $C = V^\pi(s_0)$), then B_t inflates the magnitude of $\tilde{A}_t$. It thereby can increase the variance of our policy-gradient estimate (see Appendix C). The increased variance makes the credit assignment problem harder and we therefore seek a data-dependent initialization that drives $\mathbb{E}_t[B_t] \approx 0$ over a collected batch. Intuitively speaking, our solution is to rank trajectories relative to each other which can be understood as anchoring. For a visual example, see Figure 2.

4.1 Trajectory Ranking

Given M training rollouts $\{\tau^{(m)}\}_{m=1}^{M}$ with states $\tau^{(m)} = (s_0^{(m)}, \ldots, s_T^{(m)})$ and done flags $\{d_t^{(m)}\}_{t=0}^{T-1}$ indicating whether $s_{t+1}^{(m)}$ begins a new episode. Define

$$\mathcal{K}^{(m)} := \{0\} \cup \{\, t \in \{1, \ldots, T\} : d_{t-1}^{(m)} = 1 \,\}, \tag{23}$$

as the index set where either a sub-episode or new episode starts. In the following, we refer to these selected states as *start states*. Enumerate all $N = \sum_m |\mathcal{K}^{(m)}|$ start states as $\{s_{\text{start}}^{(n)}\}_{n=1}^{N}$, where each $s_{\text{start}}^{(n)} = s_k^{(m)}$ for some $k \in \mathcal{K}^{(m)}$. Let $\Delta_\theta : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ be the learned (noisy) difference model and define the $N \times N$ matrix of pairwise differences

$$\Delta_{ij} := \Delta_\theta\Big(s_{\text{start}}^{(i)},\, s_{\text{start}}^{(j)}\Big), \qquad 1 \leq i, j \leq N. \tag{24}$$

To obtain an offset estimation that is robust to prediction noise and is identifiable up to a batchwise constant (which is sufficient for ranking), we estimate all start state offsets O via row-wise averaging and then subtract the batch minimum to get non-negative values with ranking

$$O(s_{\text{start}}^{(n)}) := \frac{1}{N} \sum_{j=1}^{N} \Delta_{nj}, \qquad \widehat{V}_\theta(s_{\text{start}}^{(n)}) := O(s_{\text{start}}^{(n)}) - \min_{1\leq \ell \leq N} O(s_{\text{start}}^{(\ell)})\,. \tag{25}$$

Finally and to avoid overly complex notation, for any state s in a rollout, use the most recent *start state* in that same rollout, call it s_{start}. Set the relative value for s as

$$\bar{V}_\theta(s) \;=\; \widehat{V}_\theta(s_{\text{start}}) + \Delta_\theta(s, s_{\text{start}}), \tag{26}$$

and use these values for R-GAE. In that way, each state gets its episode-specific offset.

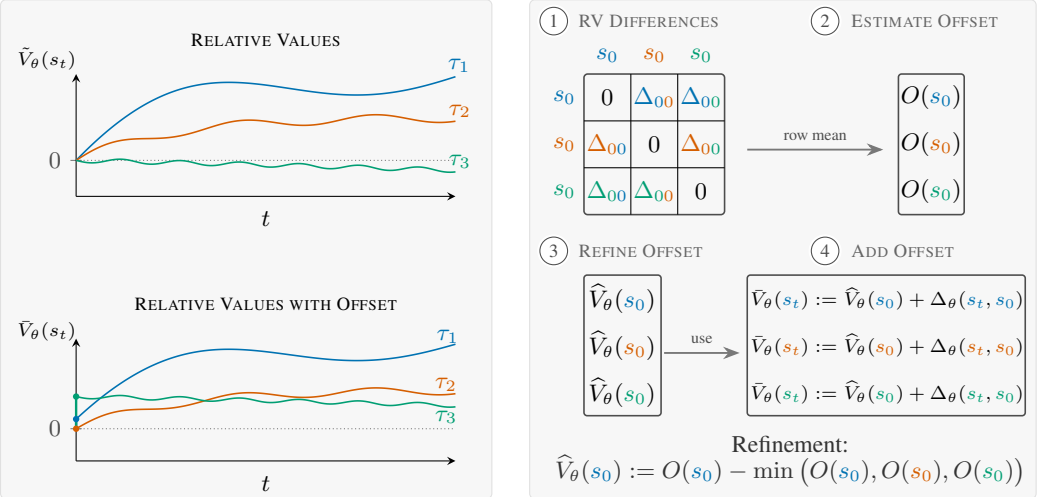

Figure 2: **Trajectory Ranking.** When training batches contain samples from more than one episode, the initialization $\tilde{V}(s_0) = 0$ for each trajectory τ_i is not correct. The trajectories need to be ranked relative to each other by adding an offset that is calculated with $\Delta(s_i, s_j)$. Note that s_0, s_0, s_0 are start states of τ_1, τ_2, τ_3 indicated by color. For simplicity, assume in this figure that start states are only present at $t = 0$ for each rollout, so we can think about each τ as one episode.

5 Training Objective

The PPO objective uses relative advantages $\tilde{A}_t$ and the usual clipped surrogate:

$$\mathcal{L}_{\text{policy}}(\theta) = \mathbb{E}_t\big[\min(r_t(\theta)\tilde{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\tilde{A}_t)\big], \quad r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\text{old}}(a_t \mid s_t)}. \tag{27}$$

The critic loss and entropy bonus are

$$\mathcal{L}_{\text{critic}}(\theta) = \tfrac{1}{2}\,\mathbb{E}_{(i,j)\sim\mu}\big[\big(\Delta_\theta(s_i, s_j) - y_{ij}^{(n)}\big)^2\big], \qquad \mathcal{L}_{\text{ent}}(\theta) = -\mathbb{E}_t[H(\pi_\theta(\cdot \mid s_t))], \tag{28}$$

where we use the n-step value target (see Eq. 21) and finally optimize the combined loss

$$\mathcal{L}(\theta) = -\mathcal{L}_{\text{policy}}(\theta) + c_v\mathcal{L}_{\text{critic}}(\theta) + c_e\mathcal{L}_{\text{ent}}(\theta). \tag{29}$$

5.1 Network Architecture

For all Atari experiments we use the exact same architecture as PPO Schulman et al. (2017). The policy and relative value function share the CNN encoder $f_{\text{enc}}(s) \in \mathbb{R}^d$ and then split their computation by using a single linear layer for their respective outputs.

Relative Critic. Relative values are obtained by applying the same encoder $f_{\text{enc}}(s)$ to both states and projecting the difference of their embeddings. We do not use an additional target encoder or stop-gradients. Formally, the value difference for a state pair (s_i, s_j) is given by

$$\Delta_\theta(s_i, s_j) = \Phi\big(f_{\text{enc}}(s_i) - f_{\text{enc}}(s_j)\big), \tag{30}$$

where Φ is the projection head. For fair comparisons, our experiments use a single learned vector $w \in \mathbb{R}^d$ without bias term to ensure antisymmetry $\Delta_\theta(s_i, s_j) = -\Delta_\theta(s_j, s_i)$ and $\Delta_\theta(s_i, s_i) = 0$ by design. It is also feasible to build Φ as an non-linear MLP that is antisymmetric in nature (e.g. by using tanh activations and no bias term in linear layers). But so far, we have not observed additional improvements by using such non-linear heads.

6 Experiments

We evaluate *Relative Value Learning* (RV) as a drop-in critic for on-policy policy-gradient methods on the Arcade Learning Environment (ALE) for Atari. Observations follow the standard PPO

preprocessing: random no-op resets, frame skip with max-over-two, grayscale resizing to 84×84, stacking $m{=}4$ frames, and input scaling to $[0, 1]$. Networks use orthogonal initialization with a small policy logit scale (0.01) and unit scale for the value head, matching widely used PPO implementations. We run with EnvPool Weng et al. (2022) for high-throughput environment simulation and train for 40M frames (10M environment steps). For each game we use 10 independent seeds and report performance as the average score over the last 100 training episodes. All hyperparameters stay identical over 49 games and are reported in Appendix D.

Table 1: **PPO+RV (ours) is competitive with PPO and DAE.** Mean final scores (last 100 episodes) with standard deviation of PPO, DAE and our method (PPO+RV) after 40 M game frames.

Game	PPO (Schulman et al., 2017)	DAE (Pan et al., 2022)	PPO + RV (ours)
Alien	1850.3±376.8	1372.7±349.3	1748.8±408.7
Amidar	674.6±108.5	394.6±121.9	504.7±108.8
Assault	4971.9±642.4	2205.1±362.3	3901.7±219.2
Asterix	4532.5±1570.7	3750.1±522.9	3862.2±678.2
Asteroids	2097.5±88.8	1392.3±62.3	1489.5±91.5
Atlantis	2 311 815.0±420555.5	2 888 011.1±225889.1	3 101 686.1±451117.9
BankHeist	1280.6±2.7	257.8±193.2	1209.0±77.5
BattleZone	17 366.7±1045.0	16 302.0±2042.5	21 780.0±1516.1
BeamRider	1590.0±276.3	1729.9±172.7	2044.0±396.3
Bowling	40.1±13.8	36.1±9.1	46.3±7.9
Boxing	94.6±0.9	25.9±9.5	94.8±0.8
Breakout	274.8±20.2	234.9±28.2	263.0±27.7
Centipede	4386.4±165.6	3915.8±694.4	1226.1±104.5
ChopperCommand	3516.3±991.6	1587.3±423.6	4435.2±965.5
CrazyClimber	110 202.0±3547.7	112 319.5±6125.4	111 622.2±5645.4
DemonAttack	11 378.4±2800.5	2477.2±344.3	8944.7±534.3
DoubleDunk	−14.9±1.3	−12.5±3.0	−23.3±5.3
Enduro	758.3±31.2	0.0±0.0	1080.2±121.1
FishingDerby	17.8±2.8	−60.4±7.2	19.8±5.2
Freeway	32.5±0.3	19.5±13.7	31.9±0.2
Frostbite	314.2±4.9	367.9±278.2	522.6±399.0
Gopher	2932.9±1870.6	1137.2±156.8	3719.1±249.5
Gravitar	737.2±150.9	443.5±50.3	1441.0±561.4
IceHockey	−4.2±0.3	−5.1±0.5	−3.9±0.4
Jamesbond	560.7±94.9	507.8±21.7	568.9±43.7
Kangaroo	9928.7±7089.9	1331.0±1060.8	5649.0±2940.0
Krull	7942.3±555.1	9034.0±774.6	8870.9±473.1
KungFuMaster	23 310.3±2251.9	20 535.3±2810.7	24 483.6±6794.6
MontezumaRevenge	42.0±57.4	0.1±0.3	1.4±3.4
MsPacman	2096.5±157.1	2501.0±600.9	1721.2±230.1
NameThisGame	6254.9±160.9	6016.3±283.3	6685.3±938.3
Pitfall	−32.9±19.0	−12.0±23.1	−27.3±26.3
Pong	20.7±0.2	20.7±0.3	16.8±3.1
PrivateEye	69.5±55.2	86.0±14.9	93.3±17.6
Qbert	14 293.3±293.1	11 119.6±3465.8	15 261.6±502.1
Riverraid	8393.6±385.4	3150.8±549.9	9465.8±1062.5
RoadRunner	25 076.0±10851.2	16 146.3±3074.0	43 346.3±6741.1
Robotank	5.5±0.7	6.9±2.7	19.5±3.1
Seaquest	1204.5±481.1	2049.8±430.3	1659.7±209.4
SpaceInvaders	942.5±266.1	914.9±219.4	712.3±134.5
StarGunner	32 689.0±949.8	5849.2±740.4	24 830.8±10990.8
Tennis	−14.8±3.8	−17.6±0.9	−31.7±3.4
TimePilot	4342.0±331.1	7252.7±1173.6	10 212.7±492.0
Tutankham	254.4±42.5	196.0±26.2	224.2±23.1
UpNDown	95 445.0±21584.1	85 438.4±19664.6	113 991.7±21370.6
Venture	0.0±0.0	0.0±0.0	5.5±15.6
VideoPinball	37 389.0±15876.6	23 958.6±3936.0	138 564.8±77425.7
WizardOfWor	4185.3±1029.4	4161.3±696.0	5084.1±522.6
Zaxxon	5008.7±1261.3	5612.2±1712.5	845.8±1567.9

Compute Resources. Each 40M-frame run completes in approximately 65 minutes on a single A100 GPU with 12 CPU cores. Across all 49 games and 10 seeds (490 runs total), this corresponds to 530 GPU-hours or 22.10 A100-days.

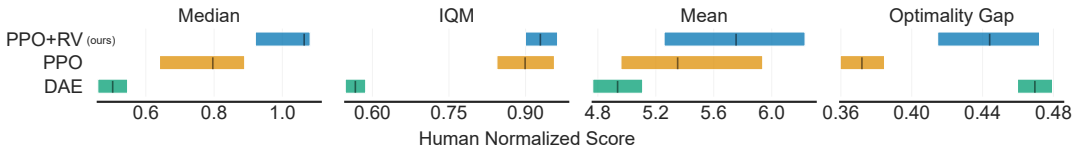

Figure 3: **Comparison between Methods.** Aggregate metrics with 95% stratified bootstrap confidence intervals (Agarwal et al., 2021). Higher median, interquartile mean (IQM), and mean, but lower optimality gap indicate better performance.

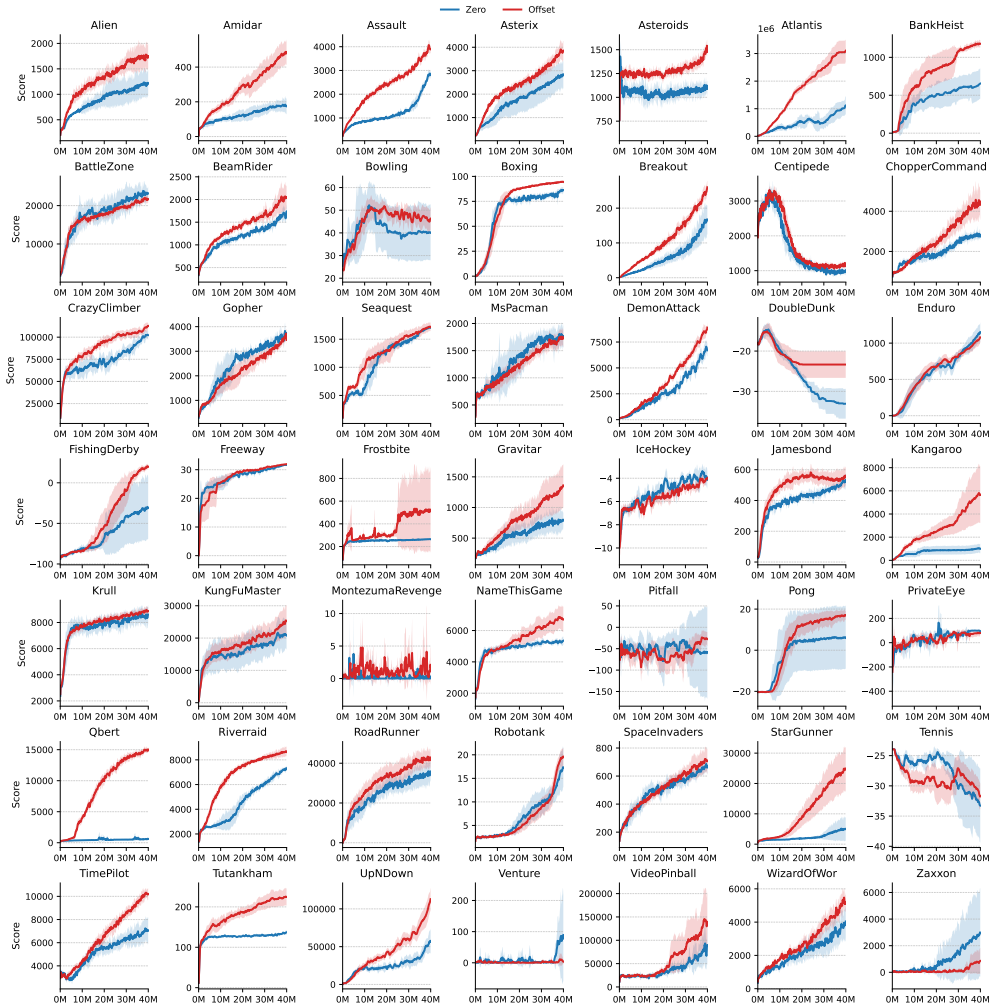

Figure 4: **Ablation for Value Initialization.** This figure compares the performance difference between zero (see Equation 9) and offset initialization (see Equation 26) for relative values. Usually the proposed offset initialization is needed to improve credit assignment, but for some games the algorithm can handle zero initialization as well.

6.1 Arcade Learning Environment (ALE)

Table 1 presents the per-game scores of PPO (Schulman et al., 2017) and Direct Advantage Estimation (DAE) (Pan et al., 2022) and extends the table with an additional column, PPO+RV. This column represents our algorithm where the absolute value critic is replaced by the RV critic. Across the benchmark, PPO+RV attains competitive performance: it exceeds PPO on 30 out of 49 games (or 61%) and DAE on 37 out of 49 games (or 75%). In addition we report aggregated metrics in Figure 3 for further analysis.

6.2 Ablation Study

Furthermore, we present the influence of relative value initialization in Figure 4. In general, we see that relative ranking is needed, but sometimes actors are not bothered by the influence of B_t which we speculate is due to the PPO clipping objective.

7 LIMITATIONS

Gauge fixing induces a trajectory-constant baseline B_t in R-GAE that inflates variance for long horizons or when $\gamma\lambda \to 1$, but it is only partially reduced by relative value initialization. The enforced antisymmetry improves stability but restricts critic expressivity (our near-linear difference head may underfit complex value differences). Pairwise training is $O(B^2)$, where B is the batch size, in the naive form and relies on subsampling that trades compute for estimator variance. Trajectory ranking used for initialization assumes post-baseline values are on a comparable scale across trajectories. Ranking signal may become uninformative. Empirically, experiments are limited to PPO on discrete-action Atari. Generality to continuous control, off-policy regimes, and preference-based settings remains to be demonstrated.

8 CONCLUSION

In this paper we established that reinforcement learning can operate on value differences rather than absolute values. We proposed *Relative Value Learning* (RV), a gauge-invariant alternative to absolute critics that learns antisymmetric value differences $\Delta^\pi(s_i, s_j) = V^\pi(s_i) - V^\pi(s_j)$ as a primitive representation. On the theoretical side, we defined a pairwise Bellman operator that is a γ-contraction on bounded antisymmetric functions with a unique fixed point equal to the true value differences, and we derived well-posed bootstrapping targets (1-step/n-step/λ) that operate entirely on observable reward differences and non terminal pairwise terms. We further introduced trajectory ranking and reconstructed generalized advantage estimation from pairwise differences (R-GAE), showing that it ensures an unbiased policy-gradient estimator and clarifying its relationship to standard GAE. Empirically, replacing the PPO critic with our relative critic gives competitive performance across Atari while simplifying the value learning objective to the quantities that matter for control: differences rather than absolutes. We hope RV serves as a building block for algorithms that reason natively in the relative domain where decisions are actually made.

9 ACKNOWLEDGMENTS

This research has been funded and supported by the Federal Ministry of Research, Technology and Space of Germany and the state of North Rhine-Westphalia as part of the Lamarr Institute for Machine Learning and Artificial Intelligence.

REFERENCES

Jinane Abounadi, Dimitrib Bertsekas, and Vivek S Borkar. Learning algorithms for markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40(3):681–698, 2001.

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

Dimitri P Bertsekas. Differential training of rollout policies. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, volume 35, pp. 913–922. UNIVERSITY OF ILLINOIS, 1997.

Dimitri P Bertsekas. Neuro-dynamic programming. In *Encyclopedia of optimization*, pp. 1–6. Springer, 2025.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 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, 2018.

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

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: A research direction. arxiv 2018. *arXiv preprint arXiv:1811.07871*, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

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 Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando De Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016a.

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, 2016b.

Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, Zhongwen Xu, and Shuicheng Yan. EnvPool: A highly parallel reinforcement learning environment execution engine. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22409–22421. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8caaf08e49ddbad6694fae067442ee21-Paper-Datasets_and_Benchmarks.pdf.

A TARGET DERIVATION FOR 1-STEP BOOTSTRAPPING

When approximating the pairwise Bellman operator from Equation 8 by using two randomly sampled transitions

$$\tau_i = (s_i, a_i, r_i, d_i, s_{i+1}), \quad \tau_j = (s_j, a_j, r_j, d_j, s_{j+1}),$$

we can encounter terminal successor states that result in ill-defined targets for $\Delta(s_{i+1}, s_{j+1})$. We present four exhaustive cases that arise from the possible combinations of terminal and non-terminal successor states where $d_i, d_j \in \{0, 1\}$ are terminal state indicators for s_{i+1} and s_{j+1}. In each case, we derive an equivalent expression for $\Delta(s_{i+1}, s_{j+1})$ by only using (1) observable rewards r_i, r_j and (2) non-terminal, well-defined value differences $\Delta_\theta(\cdot, \cdot)$. The rearrangement is then used for the bootstrapping target in Equation (20) to work for sampling.

Case 1: Both successor states are non-terminal ($d_i = 0, d_j = 0$). Then, the relative value

$$\Delta(s_{i+1}, s_{j+1}) = V(s_{i+1}) - V(s_{j+1}), \tag{31}$$

is well-posed and directly expressible with $\Delta_\theta(s_{i+1}, s_{j+1})$.

Case 2: Assume the successor s_{i+1} is terminal, and s_{j+1} is non-terminal ($d_i = 1, d_j = 0$). To address this, we derive a modified target:

$$\begin{aligned}
\Delta(s_{i+1}, s_{j+1}) &= V(s_{i+1}) - V(s_{j+1}) \\
&= V(s_{i+1}) - V(s_{j+1}) + V(s_i) - V(s_i) \\
&= \Delta(s_i, s_{j+1}) - V(s_i) + \underbrace{V(s_{i+1})}_{=0} \\
&= \Delta(s_i, s_{j+1}) - \mathbb{E}_{s_i}\left[R_t + \gamma R_{t+1} + \dots\right] \\
&\overset{\text{using } \tau_i}{\approx} \Delta(s_i, s_{j+1}) - r_i
\end{aligned} \tag{32}$$

Case 3: Assume the successor s_{i+1} is non-terminal, and is s_{j+1} terminal ($d_i = 0, d_j = 1$). To address this, we derive a modified target:

$$\begin{aligned}
\Delta(s_{i+1}, s_{j+1}) &= V(s_{i+1}) - V(s_{j+1}) \\
&= V(s_{i+1}) - V(s_{j+1}) + V(s_j) - V(s_j) \\
&= \Delta(s_{i+1}, s_j) + V(s_j) - V(s_{j+1}) \\
&= \Delta(s_{i+1}, s_j) + \mathbb{E}_{s_j}\left[R_t + \gamma R_{t+1} + \dots\right] \\
&\overset{\text{using } \tau_j}{\approx} \Delta(s_{i+1}, s_j) + r_j
\end{aligned} \tag{33}$$

Case 4: Assume that both successors are terminal ($d_i = 1, d_j = 1$). We derive a modified target:

$$\begin{aligned}
\Delta(s_{i+1}, s_{j+1}) &= V(s_{i+1}) - V(s_{j+1}) \\
&= V(s_{i+1}) - V(s_{j+1}) + V(s_i) - V(s_i) + V(s_j) - V(s_j) \\
&= \Delta(s_i, s_j) - \mathbb{E}_{s_i}\left[R_t + \gamma R_{t+1} + \dots\right] + \mathbb{E}_{s_j}\left[R_t + \gamma R_{t+1} + \dots\right] \\
&\overset{\text{using } \tau_i, \tau_j}{\approx} \Delta(s_i, s_j) - r_i + r_j
\end{aligned} \tag{34}$$

For better training stability, we simply say that

$$\begin{aligned}
\Delta(s_{i+1}, s_{j+1}) &= V(s_{i+1}) - V(s_{j+1}) \\
&= 0
\end{aligned} \tag{35}$$

since the value of both terminal states, by using MDP definition, is equal to zero.

B GAUGE FREEDOM

Lemma B.1 (Constant-offset gauge invariance)**.** *Suppose V^π satisfies the Bellman equation $V^\pi = r_\pi + \gamma P^\pi V^\pi$. For any $c \in \mathbb{R}$, define the shaped reward and shifted value*

$$r'_\pi(s) := r_\pi(s) + (1-\gamma)c, \qquad V^{\pi\prime}(s) := V^\pi(s) + c. \tag{36}$$

Then $V^{\pi\prime}$ satisfies the transformed Bellman equation

$$V^{\pi\prime} = r'_\pi + \gamma P^\pi V^{\pi\prime}. \tag{37}$$

Moreover, if one equivalently defines the reward shaping $r'(s,a) = r(s,a) + (1-\gamma)c$ so that $r'_\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[r'(s,a)]$, then action preferences and pairwise differences are invariant:

$$A^{\pi\prime}(s,a) = A^\pi(s,a) \quad \textit{and} \quad \Delta^{\pi\prime}(s_i,s_j) = \Delta^\pi(s_i,s_j) \textit{ for all } (s_i,s_j) \in S \times S. \tag{38}$$

Proof. By direct calculation,

$$\begin{aligned} r'_\pi + \gamma P^\pi V^{\pi\prime} &= \big(r_\pi + (1-\gamma)c\big) + \gamma P^\pi (V^\pi + c) \\ &= \big(r_\pi + \gamma P^\pi V^\pi\big) + \big((1-\gamma)c + \gamma c\big) \\ &= V^\pi + c \\ &= V^{\pi\prime}, \end{aligned}$$

so $V^{\pi\prime}$ satisfies the transformed Bellman equation. For invariance: with $r'(s,a) = r(s,a) + (1-\gamma)c$,

$$Q^{\pi\prime}(s,a) = r'(s,a) + \gamma\, \mathbb{E}_{s'}[V^{\pi\prime}(s')] = \big(r(s,a) + \gamma\, \mathbb{E}_{s'}[V^\pi(s')]\big) + c = Q^\pi(s,a) + c. \tag{39}$$

Thus $A^{\pi\prime}(s,a) = Q^{\pi\prime}(s,a) - V^{\pi\prime}(s) = A^\pi(s,a)$, and $\Delta^{\pi\prime}(s_i,s_j) = (V^\pi(s_i)+c) - (V^\pi(s_j)+c) = \Delta^\pi(s_i,s_j)$. □

Constant offsets are a special case of potential-based shaping (Ng et al., 1999).

C VARIANCE ANALYSIS OF THE RELATIVE POLICY GRADIENT

In Section 4, we argue that the unknown trajectory constant $|C|$ can increase the variance of the policy gradient estimator. Here, we provide the formal derivation supporting this claim and reference Figure 4 for empirical validation. First, recall from Lemma 3.2 that the relative advantage is given by $\tilde{A}_t = A_t + B_t$, where B_t is a trajectory constant determined by the initialization offset C. While Corollary 3.3 proves that the estimator remains unbiased (i.e., the first moment is unchanged), the second moment differs.

Lemma C.1 (Variance Inflation)**.** *Let $g_{std} = \nabla_\phi \log \pi_\phi(a_t|s_t) A_t$ be the standard gradient estimator and $g_{rel} = \nabla_\phi \log \pi_\phi(a_t|s_t) \tilde{A}_t$ be the relative gradient estimator. The variance of the relative estimator is given by:*

$$\mathit{Var}(g_{rel}) = \mathit{Var}(g_{std}) + \mathbb{E}\left[\|\nabla_\phi \log \pi_\phi(a_t|s_t)\|^2 B_t^2\right] + 2\mathbb{E}\left[A_t B_t \|\nabla_\phi \log \pi_\phi(a_t|s_t)\|^2\right] \tag{40}$$

Crucially, the strictly positive term $\mathbb{E}[\|\nabla_\phi \log \pi_\phi\|^2 B_t^2]$ scales quadratically with the trajectory offset C^2.

Proof. The variance of any estimator g is defined as $\mathrm{Var}(g) = \mathbb{E}[\|g\|^2] - \|\mathbb{E}[g]\|^2$. From Corollary 3.3, we know that $\mathbb{E}[g_{rel}] = \mathbb{E}[g_{std}] = \nabla_\phi J(\phi)$. Since the expected values are identical, the difference in variance is determined entirely by the second moment $\mathbb{E}[\|g\|^2]$.

Let $u_t = \nabla_\phi \log \pi_\phi(a_t|s_t)$ denote the score function. Expanding it for the second moment gives:

$$\begin{aligned} \|g_{rel}\|^2 &= \|u_t \tilde{A}_t\|^2 && (41)\\ &= \|u_t (A_t + B_t)\|^2 && (42)\\ &= \|u_t A_t\|^2 + \|u_t B_t\|^2 + 2(u_t A_t)^\top (u_t B_t) && (43) \end{aligned}$$

Note that A_t and B_t are scalars, so we can factor them out:

$$\|g_{rel}\|^2 = \underbrace{\|u_t\|^2 A_t^2}_{\|g_{std}\|^2} + \|u_t\|^2 B_t^2 + 2\|u_t\|^2 A_t B_t \tag{44}$$

Recall from Lemma 3.2 that $B_t = (1-\gamma)C \sum (\gamma\lambda)^l$. Thus, B_t is directly proportional to the initialization offset C.

1. **Noise Term:** The term $\mathbb{E}[\|u_t\|^2 B_t^2]$ is strictly non-negative. Since $B_t \propto C$, this noise term scales with C^2.
2. **Correlation Term:** The term $2\mathbb{E}[\|u_t\|^2 A_t B_t]$ represents the correlation between the true advantage and the offset. While this term can be negative, it scales linearly with C.

Consequently, for a sufficiently large uncorrected offset $|C|$, the quadratic noise term (C^2) will dominate the linear correlation term, resulting in an increase in estimator variance. This necessitates the *Trajectory Ranking* strategy (from Section 4) to minimize $|C|$ and drive $B_t \approx 0$. Empirically, our ablation in Figure 4 validates the importance of decreasing estimator variance. □

In addition to validate B_t empirically, Table 2 presents a small example for a seven step trajectory. Given are rewards and true values to compute the true GAE for comparison to relative GAE:

$$\begin{aligned} [r_0, \ldots, r_5] &:= [1, 1, 1, 1, 1, 1] \\ [V(s_0), \ldots, V(s_6)] &:= [2, 3, 4, 5, 6, 7] \end{aligned}$$

In this example $V(s_0) = 2$, so one can compute all predicted differences B_t using $C = 2$. Over all time steps, the observed advantage differences match exactly the prediction B_t, confirming the algebraic derivation from Lemma 3.2. In this example, we rounded to two decimal digits for clean visualization.

Figure 5 generalizes the intuition of Table 2 by visualizing the prediction and decay of B_t over a $T = 128$ timestep rollout, which has the same length as in our experiments. In the figure, we also

Table 2: **Empirical Evidence for the Trajectory-constant.** This tables shows a small environment trajectory and validates the trajectory-constant baseline empirically. To compute true and relative GAE, we set $\gamma = 0.9, \lambda = 0.8$ and use $C = 2$.

	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
Values							
r_t	1	1	1	1	1	1	–
$V(s_t)$	2	3	4	5	6	7	8
$\Delta(s_t, s_0)$	–	1	2	3	4	5	6
$\tilde{V}(s_t)$	0	1	2	3	4	5	6
True & Relative GAE ($\gamma = 0.9, \lambda = 0.8$)							
δ_t	1.70	1.60	1.50	1.40	1.30	1.20	–
$\tilde{\delta}_t$	1.90	1.80	1.70	1.60	1.50	1.40	–
A_t	4.73	4.21	3.63	2.96	2.16	1.20	–
$\tilde{A}_t$	5.35	4.79	4.15	3.41	2.51	1.40	–
GAE Differences ($C = 2$)							
$\tilde{A}_t - A_t$	0.61	0.58	0.52	0.45	0.34	0.20	–
Predicted via B_t	0.61	0.58	0.52	0.45	0.34	0.20	–

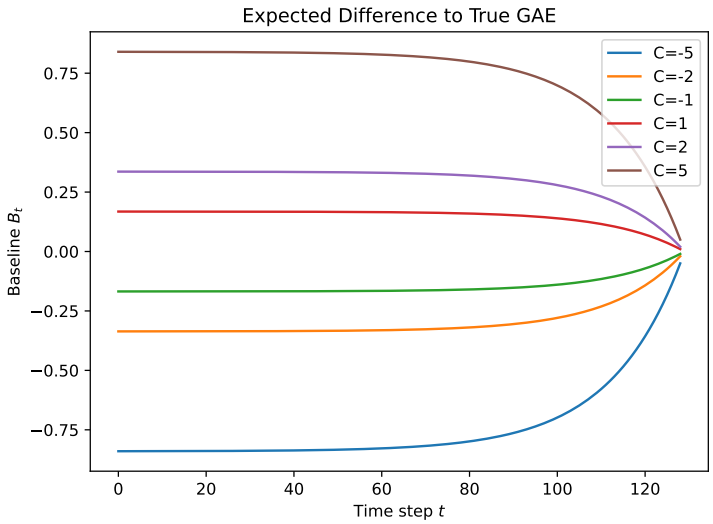

Figure 5: **Variance Visualization.** This figure shows the expected difference B_t to the true GAE A_t using different baseline values C. Note, that the difference is higher for larger $|C|$, because γ applies a stronger absolute discount.

consider the same hyperparameter setting ($\gamma = 0.99, \lambda = 0.95$) which influences the magnitude of B_t. At the start $t = 0$ of a rollout, one can see the absolute value of B_0 is large but almost constant, and inflates the GAE value. Then after some time, B_t falls off and finally ($t \rightarrow$128) almost vanishes, so R-GAE $\approx$ GAE for the last few timesteps. This can be demonstrated for arbitrary C.

D HYPERPARAMETERS

We present the hyperparameter configuration for the Atari benchmark (ALE) which is used for all 49 games.

Table 3: Hyperparameters for PPO+RV used in our experiments.

Hyperparameter	**Value**
Discount factor γ	0.99
GAE parameter λ	0.95
N-step return target	5
Clip parameter ϵ	0.1
Number of epochs per update	5
Minibatch size	128
Number of parallel environments	8
Rollout length T	128
Learning rate	2.5×10^{-4}
Optimizer	Adam
Adam epsilon	1×10^{-5}
Entropy coefficient c_e	0.01
RV loss coefficient c_v	1.25
RV value clipping	0.15
Gradient clipping (max-norm)	0.5

Additional Hyperparameters Details. For training the relative critic we do not use purely random state pairing. With probability 33%, the second state is chosen from the same episode as the reference state s_i, encouraging temporally coherent comparisons. Otherwise, the partner is sampled randomly in the batch. See Appendix E for an ablation study.

E ABLATION ON PAIR SAMPLING

In this section, we present Table 4 as an ablation study for different pair sampling strategies μ that can be applied for Equation 28. In our case, "Biased" means that with probability $p \in [0, 1]$ the second state s_j is randomly sampled from the same episode. In general, we can see that the sampling strategy has a rather small influence except when p gets too large. Then there is a small decrease in performance.

Table 4: **Ablation of Pair Sampling.** Mean final scores (last 100 episodes) with standard deviation of our method PPO+RV with different pair sampling strategies after 40 M game frames. The results average over five seeds.

Game	Random	Biased ($p = 0.33$)	Biased ($p = 0.66$)
Alien	2123.4±346.0	1746.5±491.7	1802.0±600.5
Assault	4214.3±338.3	4018.8±174.1	3615.2±290.2
Asterix	4303.2±1024.8	4495.2±1111.0	3821.2±461.1
Atlantis	2 683 819.6±515071.2	2 991 307.2±542845.7	3 711 740.4±431880.2
BattleZone	22 512.0±1038.6	22 576.0±921.1	24 616.0±1228.4
BeamRider	2199.2±573.9	1966.5±130.1	1814.8±247.2
Breakout	237.0±18.5	258.5±31.6	241.7±7.6
DemonAttack	9364.3±1069.8	8669.9±734.8	7181.1±858.8
Gopher	3224.7±1097.3	3585.7±183.1	3626.2±338.1
Gravitar	1414.2±759.3	1379.0±488.9	1232.0±406.9
IceHockey	−3.7±0.6	−3.7±0.6	−3.9±0.4
Krull	9048.9±729.2	8527.1±393.7	8540.5±504.5
MsPacman	1640.0±147.4	1783.4±125.8	1680.9±447.8
NameThisGame	7000.3±1446.8	6880.2±937.0	5890.7±296.8
Qbert	16 630.4±938.0	15 472.7±417.3	15 631.9±789.1
Riverraid	9941.6±1440.7	9271.0±1276.2	8717.8±478.7
Robotank	22.5±1.0	21.3±2.0	19.7±1.5
Seaquest	1592.6±307.4	1612.4±109.6	1236.1±321.7
TimePilot	9701.6±435.3	9825.6±819.3	9692.8±584.0
UpNDown	66 393.6±27590.7	101 955.1±25061.7	100 891.3±33878.0
VideoPinball	144 597.1±89747.6	110 187.1±57998.8	82 002.4±16130.5
WizardOfWor	5501.2±787.0	4890.0±494.9	5128.0±374.3

F THE USE OF LARGE LANGUAGE MODELS (LLMS)

Finally, we report on the use of LLMs in this section. The image in Figure 1 is generated with Gemini 2.5 Flash Image (Nano Banana). Furthermore, we used large language models (LLMs) as writing assistants in the preparation of this manuscript. LLMs were employed to draft and refine text as well as to generate preliminary versions of some proof derivations. All mathematical results, derivations, and theoretical claims were independently verified and validated by the authors.