Emphatic Algorithms for Deep Reinforcement Learning

Ray Jiang [1] **Tom Zahavy** [1] **Zhongwen Xu** [1] **Adam White** [1 2]
Matteo Hessel [1] **Charles Blundell** [1] **Hado van Hasselt** [1]

Abstract

Off-policy learning allows us to learn about possible policies of behavior from experience generated by a different behavior policy. Temporal difference (TD) learning algorithms can become unstable when combined with function approximation and off-policy sampling—this is known as the "deadly triad". Emphatic temporal difference (ETD(λ)) algorithm ensures convergence in the linear case by appropriately weighting the TD(λ) updates. In this paper, we extend the use of emphatic methods to deep reinforcement learning agents. We show that naively adapting ETD(λ) to popular deep reinforcement learning algorithms, which use forward view multi-step returns, results in poor performance. We then derive new emphatic algorithms for use in the context of such algorithms, and we demonstrate that they provide noticeable benefits in small problems designed to highlight the instability of TD methods. Finally, we observed improved performance when applying these algorithms at scale on classic Atari games from the Arcade Learning Environment.

Off-policy learning, whereby an agent learns from behavior that differs from its current policy, affords an agent opportunities to accumulate rich knowledge (Degris & Modayil, 2012) by learning about the effect of different policies of behaviors. This can also be extended to learn about different goals, e.g., by learning *general value functions* (Sutton et al., 2011) for cumulants that differ from the main task reward. Unfortunately, it is well known that reinforcement learning algorithms (Sutton & Barto, 2018) can become unstable when combining function approximation, off-policy learning, and bootstrapping (Tsitsiklis & Van Roy, 1997)—for this reason such combination is referred to as the *deadly triad* (Sutton & Barto, 2018; van Hasselt et al., 2018).

[1]DeepMind, London, UK. [2]Amii, Department of Computing Science, University of Alberta. Correspondence to: Ray Jiang <rayjiang@google.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

Many reinforcement learning (RL) agents learn off-policy to some extent, to learn about the greedy policy while exploring (Watkins, 1989), to make predictions about policies simultaneously (Sutton et al., 2011; Zahavy et al., 2020; Jaderberg et al., 2017), to improve sample complexity via experience replay (Lin, 1992; Mnih et al., 2015), or even just to correct for the latency introduced by distributed computation (Espeholt et al., 2018). Since these algorithms make use of bootstrapping and function approximation, they may suffer from deadly triad symptoms of "soft divergence" and slow convergence (van Hasselt et al., 2018).

The ETD(λ) algorithm (Sutton et al., 2016) ensures convergence with linear function approximation (Yu, 2015) by weighting the updates of TD(λ) (in the backward view, with eligibility traces). However, combining eligibility traces and deep neural networks can be challenging (Sutton, 1987)[1], and thus widely used deep RL systems instead typically use n-step forward view methods. Overall, none of the existing solutions to the deadly triad (Sutton et al., 2009; 2016) have become standard practice in deep RL. In this paper, we extend the emphatic method to multi-step deep RL learning targets, including an off-policy value-learning method known as 'V-trace' (Espeholt et al., 2018) that is often used in actor-critic systems.

The structure of this paper is the following. Sec. 1 explains the background on forward view learning targets and ETD(λ). Next we adapt ETD(λ) to the forward view in Sec. 2.1, and derive a new multi-step emphatic trace for n-step TD in Sec. 2.2. We discuss further algorithmic considerations in Sec. 2.3, including extensions for variance reduction, for the V-trace value learning target and for the actor critic learning algorithms. Empirically, we provide an in-depth comparison of their qualitative properties on small diagnostic MDPs in Sec. 3. Finally, we demonstrate that combining emphatic trace with deep neural networks can improve performance on classic Atari video games in Sec. 4, reporting the highest score to date for an RL agent without experience replay in the 200M frames data regime: 497% median human normalized score across 57 games, improved from the baseline performance of 403%.

[1]See van Hasselt et al. (2021) for recent developments.

1. Background

A Markov decision process (MDP; Bellman, 1957) consists of finite sets of states $\mathcal{S}$ and actions $\mathcal{A}$, a reward function $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, a transition distribution $P(s'|s, a)$ $s, s' \in \mathcal{S}, a \in \mathcal{A}$, and a discount factor γ. A policy is a distribution over actions conditioned on the state: $\pi(a|s)$. The goal of RL is to find a policy π that maximizes the expected discounted *return* $v_\pi(s) \doteq \mathbb{E}_\pi \left[\sum_{t\geq 0}(\prod_{i=1}^{t}\gamma_t)R_{t+1}\right]$ where, at time t, γ_t denotes the scalar discount, $S_t \in \mathcal{S}$ the state variable, $A_t \in \mathcal{A}$ the action taken and $R_{t+1} \doteq r(S_t, A_t)$ the reward.[2]

1.1. TD(λ)

Policy *evaluation* is the problem of learning to predict the value $V_{\boldsymbol{\theta}}(s) \approx v_\pi(s)$, for all states s, under an arbitrary (fixed) policy π and parametrized by $\boldsymbol{\theta}$. When using function approximation, each state S_t is associated with a feature vector $\boldsymbol{\phi}_t$, and the agent's value estimates $V_{\boldsymbol{\theta}}(s)$ are a parametric function of these features. TD(λ) (Sutton, 1988) is a widely used algorithm for policy evaluation where, on each step t, the parameters of $V_{\boldsymbol{\theta}}$ are updated according to

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha_t \delta_t \mathbf{e}_t \,,$$

where $\mathbf{e}_t = \gamma_t \lambda \mathbf{e}_{t-1} + \nabla_{\boldsymbol{\theta}} V_{\boldsymbol{\theta}}(S_t)$ is an *eligibility trace* of value gradients, $\delta_t = R_{t+1} + \gamma v_{\boldsymbol{\theta}}(S_{t+1}) - V_{\boldsymbol{\theta}}(S_t)$ is the *temporal difference (TD) error*, and $\alpha_t \in [0, 1]$ is the step-size. With linear function approximation $V_{\boldsymbol{\theta}}(t) = \boldsymbol{\theta}^\top \boldsymbol{\phi}_t$, the gradient $\nabla_{\boldsymbol{\theta}} V_{\boldsymbol{\theta}}(S_t)$ is the state features $\boldsymbol{\phi}_t$. TD(λ) uses *bootstrapping*, where the agent's own value estimates $V_{\boldsymbol{\theta}}(S_t)$ are used to update the values online, on each step, without waiting for the episodes to fully resolve. TD algorithms can also be used to learn policies (i.e. for *control*), by using similar updates to learn action values, or by combining value learning with policy gradients in actor-critic systems (Sutton et al., 2000).

Temporal difference algorithms can be extended to policy evaluation (or control) in *off-policy* settings, where the agent learns predictions about a *target* policy π, from trajectories $(S_i, A_i, R_{i+1})_{i=t}^{t+n}$ sampled under a different *behavior* policy μ. However, when combining function approximation with bootstrapping and off-policy learning, the parameters may diverge (Baird, 1995; Tsitsiklis & Van Roy, 1997), a phenomenon referred to as the *deadly triad*.

1.2. Emphatic TD(λ)

Emphatic TD(λ) (Sutton et al., 2016) resolves the instability due to the deadly triad by adjusting the magnitude of updates on each time-step. The idea is to re-weight the distribution of TD(λ) updates to account for the likelihood of the trajectory leading to the updated state, under the target policy. Each update is emphasized or de-emphasized by a scalar *follow-on trace*[3]:

$$F_t \doteq \gamma(S_t)\rho_{t-1}F_{t-1} + 1. \tag{1}$$

The Emphatic TD(λ) algorithm (Sutton et al., 2016), ETD(λ) for short, incorporates F_t into the conventional eligibility trace update of TD(λ) by emphasizing states

$$e_t \doteq \rho_t\big(\gamma(S_t)\lambda(S_t)e_{t-1} + M_t\boldsymbol{\phi}_t\big)\,.$$

where $\rho_t \doteq \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}$ is the *importance sampling (IS)* ratio for the target policy π and the behavior policy μ. The *emphatic trace* M_t encodes how much the current state is bootstrapped by other states based on the follow-on trace:

$$M_t = \lambda(S_t) + (1 - \lambda(S_t))F_t\,. \tag{2}$$

Prior work on off-policy TD(λ) corrected the state-distribution using the stationary distribution induced by the target policy (Precup et al., 2001), unlike ETD(λ) which uses the distribution of states produced by starting the target policy in the stationary distribution of the behavior policy.

Extensions to ETD(λ) include the ETD(λ, β) algorithm, which uses an additional hyper-parameter β in place of the discount γ to control variance by setting $\beta < \gamma$ (Hallak et al., 2016), and the ACE algorithm that applies emphatic weightings to policy gradient updates (Imani et al., 2018).

ETD(λ) is convergent with *linear* function approximation (Yu, 2015), but its performance when combined with *non-linear* function approximation has not yet been extensively evaluated.

1.3. n-step TD

In this paper, we generalize the emphatic approach to widely used deep RL systems, and in particular actor-critic systems. Contrasting with the backward view TD(λ) learning target for which ETD(λ) was developed, deep RL algorithms are often based on a *forward view* of temporal difference learning, where updates are computed on trajectories of fixed length, without making use of eligibility traces.

If we use a linear value function parametrized by $\boldsymbol{\theta}$, then the n-step TD update for parameters $\boldsymbol{\theta}$ in the first state is

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \sum_{i=t}^{t+n-1} \Big(\prod_{j=t}^{i-1} \rho_j \gamma_{j+1}\Big)\, \rho_i \delta_i(\boldsymbol{\theta}_t)\boldsymbol{\phi}_t, \tag{3}$$

where

$$\delta_i(\boldsymbol{\theta}_t) = R_{i+1} + \gamma_{i+1} V_{\boldsymbol{\theta}_t}(S_{i+1}) - V_{\boldsymbol{\theta}_t}(S_i)\,. \tag{4}$$

[2] We use the notation "$\doteq$" to indicate an equality by definition rather than by derivation.

[3] The original formula $F_t^e \doteq \gamma(S_t)\rho_{t-1}F_{t-1}^e + i_t$ has an additional scalar i_t, indicating "interest" in state S_t. We let $i_t \doteq 1$.

n-step TD can be implemented in a computationally efficient way where multiple states in a trajectory are updated at once. This can be done in two ways. In a *fixed* update scheme all states are updated with n step TD target for a fixed constant n. In a *mixed* update scheme, the k-th sample in the trajectory uses an $(n-k)$-step TD target—this is convenient when we used a small batch of temporal data, and all returns bootstrap on the last available state at the end of this window.

1.4. V-trace

Given a trajectory of data, sampled from behavior policy μ, the n-step V-trace estimator can be used as target to learn the value of state S_t under the target policy π. Let G_t be the V-trace target:

$$G_t \doteq V_{\boldsymbol{\theta}_t}(S_t) + \sum_{i=t}^{t+n-1} \Big(\prod_{j=t}^{i-1} \bar{c}_j \gamma_{j+1}\Big)\, \bar{\rho}_i \delta_i(\boldsymbol{\theta}_t), \tag{5}$$

where $\bar{\rho}_i \doteq \min(\bar{\rho}, \frac{\pi(A_i|S_i)}{\mu(A_i|S_i)})$, $\bar{c}_j \doteq \min(\bar{c}, \frac{\pi(A_j|S_j)}{\mu(A_j|S_j)})$. The clipping hyper-parameters $\bar{\rho}$ and $\bar{c}$ were introduced to reduce variance of the n-step off-policy TD target. In practice, the clipping thresholds $\bar{c}$ and $\bar{\rho}$ are often equal, so that $\bar{c}_t = \bar{\rho}_t$.

Modifying $\bar{c}$ does not change the fixed point of the (tabular) V-trace update (see the proof of the V-trace fixed point in Appendix A of Espeholt et al. (2018), and see also Mahmood et al. (2017)). Modifying $\bar{\rho}$ does change the fixed point, which corresponds to the value of the following policy $\pi_{\bar{\rho}}$:

$$\pi_{\bar{\rho}}(a|s) \doteq \frac{\min(\bar{\rho}\mu(a|s), \pi(a|s))}{\sum_{a' \in \mathcal{A}} \min(\bar{\rho}\mu(a'|s), \pi(a'|s))}. \tag{6}$$

With linear functions the V-trace update closely matches (3), except in clipping all IS weights.

1.5. Actor-critics

The V-trace update is most often used in actor-critic systems. Here, in addition to using it for learning values (the *critic*) we can use the V-trace target also in the policy update.

Consider a current policy $\pi_{\mathbf{w}}$. Following the derivation of policy gradient in Espeholt et al. (2018), we may update policy parameters $\mathbf{w}$ in the direction of the policy gradient

$$\bar{\rho}_t (R_{t+1} + \gamma_{t+1} G_{t+1} - V_{\boldsymbol{\theta}}(S_t)) \nabla_{\mathbf{w}} \log \pi_{\mathbf{w}}(A_t|S_t)\,, \tag{7}$$

where G_{t+1} is the V-trace value target from time step $t+1$ onward. This has been very successful (e.g., Espeholt et al., 2018; Hessel et al., 2019) in setting where the off-policyness is mild.

2. Proposed Emphatic Methods

Similar to TD(λ), off-policy n-step TD can suffer from unstable learning due to the deadly triad. In the appendix, we analyze the update and derive conditions that guarantee stable learning when the behavior policy is sufficiently close to the target policy. However, these conditions are often violated in practice when the policies are too different. Then emphatic methods could help stabilize learning by mitigating the mismatch in steady-state distributions under the target and behavior policies. Therefore, we now first adapt ETD(λ) to make use of n-step targets and analyze its properties, and then introduce new updates that combine emphatic methods with off-policy targets based on V-trace.

2.1. Windowed ETD(λ) — WETD

As described in Sec. 1.2, ETD(λ) uses TD(λ) as its learning target. To extend this idea, we adapt ETD(λ) to use update windows of length n. Each state in the window is updated with a variable bootstrap length, all bootstrapping on the last state in the window — this is the mixed update scheme. We formulate the learning target TD(λ) as a *mixed n-step target* by setting λ_t to 0 every n steps:

$$\lambda_t \doteq \begin{cases} 0, & \text{if } t \bmod n = 0. \\ 1, & \text{otherwise.} \end{cases} \tag{8}$$

The bootstrapping step of the update is at the nearest future time step that is a multiple of the window size n. We set λ to 0 at the end of every update window canceling all future TD errors from that point onward. More details are provided in the appendix. ETD(λ), as in (2), was originally derived for state-dependent $\lambda(S_t)$. To apply the same derivation to a time-dependent λ_t, we note that under mild assumptions (that state visits are non-periodic), the value of λ_t in (8) is statistically independent of the state. This means the expected updates are asymptotically equivalent to using a uniform $\lambda = \mathbb{E}_{d_\mu}[\lambda_t] = 1 - 1/n$.

The *windowed ETD(λ)* (WETD) algorithm is then defined by using λ_t from (8) in the ETD(λ) update in (2), so that

$$M_t^w \doteq \lambda_t + (1 - \lambda_t) F_t\,, \tag{9}$$

with (1) and (8). The WETD-corrected n-step TD target is obtained by multiplying M_t^w to weight each update:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha M_t^w \sum_{i=t}^{t+n-1} \Big(\prod_{j=t}^{i-1} \gamma_{j+1} \rho_j\Big)\, \rho_i \delta_i(\boldsymbol{\theta}_t) \boldsymbol{\phi}_t\,. \tag{10}$$

The algorithm is shown below in Algo. 1.

2.2. Emphatic TD(n) — NETD

We also investigate the use of off-policy n-step TD target and derive from scratch a new emphatic trace called *Emphatic TD(n)*, abbreviated as *NETD*. Similar to ETD(λ), NETD guarantees asymptotic stability in off-policy learning with linear value function approximation by ensuring that

Algorithm 1 WETD weighted n-step TD.

Input: Target policy π, behavior policy μ, bootstrapping length n, gradient step size α, discounts γ_t, state features $\boldsymbol{\phi}_t$.
Initialize model parameters $\boldsymbol{\theta}_0$.
Initialize $F_0 = 1$.
for $t \in [0, n, 2n, \dots, Ln]$ **do**
  Sample trajectory $(S_i, A_i, R_{i+1})_{i=t}^{t+n} \sim \mu$.
  Set $\rho_i = \pi(A_i|S_i)/\mu(A_i|S_i)$, for $i = t, \dots, t+n$.
  for $k \in [0, \dots, n-1]$ **do**
    Compute $F_{t+k+1} = \gamma_{t+k+1}\rho_{t+k}F_{t+k} + 1$.
    if $k = 0$ **then** $M_{t+k}^w = F_{t+k}$ **else** $M_{t+k}^w = 1$ **end**
    Update model parameters:
    $\boldsymbol{\theta}_{t+k+1} = \boldsymbol{\theta}_{t+k} +$
    $\alpha M_{t+k}^w \sum_{i=t+k}^{t+n-1} (\prod_{j=t+k}^{i-1} \gamma_{j+1}\rho_j)\ \rho_i \delta_i(\boldsymbol{\theta}_{t+k})\boldsymbol{\phi}_{t+k}$.
  end
end
Return: $\boldsymbol{\theta}_{Ln}$.

Algorithm 2 NETD weighted n-step TD.

Input: Target policy π, behavior policy μ, bootstrapping length n, gradient step size α, discounts γ_t, state features $\boldsymbol{\phi}_t$.
Initialize model parameters $\boldsymbol{\theta}_0$.
Initialize $F_0^{(n)}, \dots, F_{n-1}^{(n)} = 1$.
Sample trajectory $(S_i, A_i, R_{i+1})_{i=0}^{n-1} \sim \mu$.
Set $\rho_i = \pi(A_i|S_i)/\mu(A_i|S_i)$, for $i = 1, \dots, n-1$.
for $t \in [0, \dots, T]$ **do**
  Sample $S_{t+n}, A_{t+n}, R_{t+n+1} \sim \mu$
  Set $\rho_{t+n} = \pi(A_{t+n}|S_{t+n})/\mu(A_{t+n}|S_{t+n})$.
  if $t \geq n$ **then**
  Compute $F_t^{(n)} = \prod_{i=1}^{n}(\gamma_{t-i+1}\rho_{t-i})F_{t-n}^{(n)} + 1$.
  end
  Update model parameters:
  $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha F_t^{(n)} \sum_{i=t}^{t+n-1} (\prod_{j=t}^{i-1} \gamma_{j+1}\rho_j)\ \rho_i \delta_i(\boldsymbol{\theta}_t)\boldsymbol{\phi}_t$.
end
Return: $\boldsymbol{\theta}_T$.

the asymptotic update matrix is positive definite (see the appendix for its derivation and stability analysis).

Consider an n-step TD update (in the fixed update scheme). We define the NETD trace as

$$F_t^{(n)} = \prod_{i=1}^{n}(\gamma_{t-i+1}\rho_{t-i})F_{t-n}^{(n)} + 1, \tag{11}$$

where $F_0^{(n)}, F_1^{(n)}, \dots, F_{n-1}^{(n)} = 1$. We can apply this new trace to weight each n-step TD update to $\boldsymbol{\theta}$, i.e.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha F_t^{(n)} \sum_{i=t}^{t+n-1} (\prod_{j=t}^{i-1} \gamma_{j+1}\rho_j)\ \rho_i \delta_i(\boldsymbol{\theta}_t)\boldsymbol{\phi}_t\,. \tag{12}$$

NETD accumulates every n steps, making it a tamer trace than the WETD follow-on trace F_t (see Prop. 1). For a concrete example, consider $\gamma \equiv 0.99$ and in the on-policy case, $\rho \equiv 1$. For WETD, the fixed point of F_t is 100, whereas the fixed point of $F_t^{(n)}$ is $1/(1-0.99^n)$, which is 10.46 for $n = 10$, 3.84 for $n = 30$, and only 1.58 for $n = 100$. As a result, NETD can be more stable than WETD when large bootstrap lengths are used, which is common in practice.

Proposition 1. *Assume $\rho_t > 0$ and $\gamma_t > 0$ for any time step t. Then we have $F_t > F_t^{(n)}$ for any $t > 0$.*

Proof. We prove this result by induction for every n time steps. At the start, for $t = 0$, we have $F_0 = F_0^{(n)} = 1$. For $0 < t \leq n$, since $\rho_t > 0, \gamma_t > 0$ and $F_0 = 1$, we know $F_t = \gamma_t\rho_{t-1}F_{t-1} + 1$ is always a positive number. Thus $F_t = \gamma_t\rho_{t-1}F_{t-1} + 1 > 1 = F_t^{(n)}$ since $\gamma_t\rho_{t-1}F_{t-1} > 0$.

Now assume that $F_{t-n} > F_{t-n}^{(n)}$, we derive that $F_t > F_t^{(n)}$ as follows. Substituting Eq. (1) n times, we have

$$F_t = \prod_{i=1}^{n}(\gamma_{t-i+1}\rho_{t-i})F_{t-n} \tag{13}$$

$$+ \sum_{k=1}^{n-1}\prod_{j=1}^{k}(\gamma_{t+1-j}\rho_{t-j}) + 1 \tag{14}$$

$$\geq F_t^{(n)} + \sum_{k=1}^{n-1}\prod_{j=1}^{k}(\gamma_{t+1-j}\rho_{t-j}) > F_t^{(n)}. \tag{15}$$

Since $F_t > F_t^{(n)}$ for $t = 0, 1, \dots, n$, this is also true for $t = n+1, \dots, 2n+1$ and so on for every time step t. □

Therefore F_t used in WETD is a strict upper bound for $F_t^{(n)}$ in NETD. Moreover the difference between them $\sum_{k=1}^{n-1}\prod_{j=1}^{k}(\gamma_{t+1-j}\rho_{t-j})$ grows with n. Algo. 2 shows the pseudo-code of NETD weighted TD learning.

2.3. Emphatic Variants

In this section we derive novel emphatic updates based on on either the WETD or the NETD trace.

Clipped Emphases To further reduce variance of the emphatic algorithms, we clip the IS weights used in computing WETD and NETD in Eq. 9 & 11, and keep the IS weights used in computing the learning update unchanged. We call this new emphatic trace *Clip-WETD* in the case of WETD,

$$\bar{F}_t = \bar{\rho}_{t-1}\gamma_t\bar{F}_{t-1} + 1, \tag{16}$$

	Emphatic Trace Computation		Learning Target Computation			
Algorithm	transform ρ	trace type	learning target	update scheme	clip c	clip ρ
n-step TD[1]	N/A	x	π^*	either	x	x
NETD	x	NETD	π^*	fixed	x	x
WETD	x	WETD	π^*	mixed	x	x
Clip-NETD	$\min(\bar{\rho}, \rho)$	NETD	unknown	fixed	x	x
Clip-WETD	$\min(\bar{\rho}, \rho)$	WETD	unknown	mixed	x	x
V-trace[1]	N/A	x	$\pi_{\bar{\rho}}$	either	✓	✓
NEVtrace	$\rho^v \doteq \pi_{\bar{\rho}}/\mu$	NETD	$\pi_{\bar{\rho}}$	fixed	✓	✓
WEVtrace	$\rho^v \doteq \pi_{\bar{\rho}}/\mu$	WETD	$\pi_{\bar{\rho}}$	mixed	✓	✓

Table 1. Look-up table for our emphatic algorithms and the two baseline algorithms without emphatic traces. π^* is the optimal policy for n-step TD learning. $\pi_{\bar{\rho}}$ is the fixed point target policy of V-trace (Eq. 6). When applied to Surreal (explained in Sec. 4) for large scale experiments, we always clip IS weights in computing emphatic traces to reduce variance except for NEVtrace and WEVtrace.

and *Clip-NETD* in the case of NETD,

$$\bar{F}_t^{(n)} = \prod_{i=1}^{n} (\gamma_{t-i+1}\bar{\rho}_{t-i})\bar{F}_{t-n}^{(n)} + 1. \tag{17}$$

Note that clipping reduces the growth of emphatic traces, but may introduce bias in the emphatic trace weighted learning updates.

Emphatic V-trace We can also use the emphatic traces WETD and NETD in combination with V-trace value target. In the case of WETD, we first adapt TD(λ) to the mixed V-trace learning target with windows of length n by defining the new λ_t^v as

$$\lambda_t^v \doteq \begin{cases} 0, & \text{if } t \bmod n = 0. \\ \bar{\rho}_t/\rho_t, & \text{otherwise.} \end{cases} \tag{18}$$

where $\bar{\rho}_t = \min(\bar{\rho}, \rho_t)$ and $\bar{\rho}$ is the clipping threshold on IS weights of the learning target. Similar to the adaption to off-policy n-step TD target, this way the future TD errors not only stop affecting the update if they lie beyond the current window (due to λ^v set to 0), but also the relevant TD errors are weighted according to the clipped IS weights as in the V-trace value target. The appendix contains a detailed analysis on why this recovers the mixed V-trace learning target. We then adapt WETD to the V-trace learning target by adopting its target policy ($\pi_{\bar{\rho}}$ in Eq. 6) as the target policy of the emphatic trace. The IS ratios in computing F_t^v are between the V-trace target policy $\pi_{\bar{\rho}}$ and the behavior policy μ, i.e. $\rho_t^v = \pi_{\bar{\rho}}(A_t|S_t)/\mu(A_t|S_t)$, and

$$F_t^v = \rho_{t-1}^v \gamma_t F_{t-1}^v + 1. \tag{19}$$

We call this new emphatic trace the Windowed Emphatic Vtrace, abbreviated as *WEVtrace*.

In the case of NETD, we similarly extend it to the fixed V-trace target by replacing the IS weights in Eq 11 by ρ_t^v. We call it the N-step Emphatic V-trace, *NEVtrace*.

$$F_t^{(n),v} = \prod_{i=1}^{n} (\gamma_{t-i+1}\rho_{t-i}^v)F_{t-n}^{(n),v} + 1. \tag{20}$$

See the appendix for derivation details. Notice that NEVtrace and WEVtrace are likely to have higher variances than Clip-NETD and Clip-WETD (see the inequality below for any time step $t > 0$). Though they are the correct emphatic traces w.r.t. to the V-trace target, in practice they often perform worse than the clipped emphatic traces when applied to the V-trace target.

$$\rho_t^v = \pi_{\bar{\rho}}(A_t|S_t)/\mu(A_t|S_t) \tag{21}$$

$$= \frac{\min(\bar{\rho}, \pi(A_t|S_t)/\mu(A_t|S_t))}{\sum_{a' \in \mathcal{A}} \min(\bar{\rho}\mu(a'|A_t), \pi(a'|S_t))} \tag{22}$$

$$= \frac{\bar{\rho}_t}{\sum_{a' \in \mathcal{A}} \min(\bar{\rho}\mu(a'|S_t), \pi(a'|S_t))} \geq \bar{\rho}_t. \tag{23}$$

Table 1 lists all the emphatic algorithms and the three baseline learning algorithms with their respective variations in learning updates and emphatic trace computation.

Emphatic Actor-critics Actor critic agents reportedly can suffer more from off-policy learning than value-based agents, which is one of the main reasons we choose to focus on V-trace in this paper. We can combine the emphatic traces derived above with the off-policy n-step TD or the V-trace value targets, and apply these to actor critic by simply applying emphatic traces to both the value estimate gradient and the policy gradient in for example, the V-trace learning update, following existing work on ACE (Imani et al., 2018). We name these new emphatic algorithms after the emphatic trace used, which can be any of, e.g. NETD, WETD, Clip-WETD, Clip-NETD, NEVtrace, WEVtrace. We add an '-ACE' suffix to indicate when the same emphatic trace is applied not just to the value update, but also to the policy gradient update.

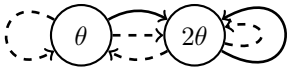


Figure 1. A simple two State MDP. The solid lines depict the (deterministic) target policy $\pi(\text{right}|\cdot) = 1$. Dashed lines denote the (more exploratory) behavior policy μ. The behavior policy selects any of the actions with equal probability $\mu(\text{right}|\cdot) = \mu(\text{left}|\cdot) = 0.5$, in all states. The rewards are zero everywhere.

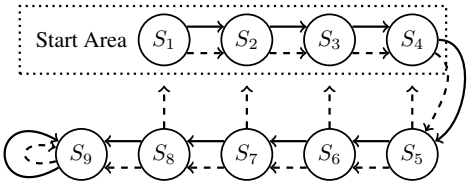


Figure 2. Collision Problem. Solid lines depict target policy $\pi(\text{forward}|\cdot) = 1$. Dashed lines indicate the behavior policy $\mu(\text{forward}|x \in \mathcal{S}_{\text{start}} \cup S_9) = 1, \mu(\text{forward}|x \in \mathcal{S}_{\text{next}}) = 0.5$, where $\mathcal{S}_{\text{start}} = \{S_1, S_2, S_3, S_4\}$ and $\mathcal{S}_{\text{next}} = \{S_5, S_6, S_7, S_8\}$. On a *retreat* action, the agent goes back to a random state in $\mathcal{S}_{\text{start}}$.

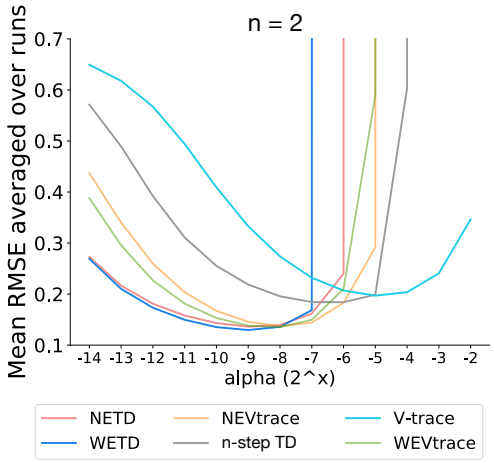


Figure 3. Hyper-parameter sensitivity comparison on the Collision problem. Each data point in the plot shows the mean RMSE averaged over 200 runs for different values of learning rate α. The emphatic algorithms achieve best performance in this task, and do so with smaller step-sizes than V-trace and n-step TD; consistent with previous results on this task (Ghiassian et al., 2018)

3. Diagnostic Experiments

We empirically analyze the properties of these new emphatic algorithms to observe how qualitative properties such as convergence, learning speed and variance manifest in practice. We examine these in the context of two small scale diagnostic off-policy policy evaluation problems: (1) a two-state MDP, shown in Figure 1, commonly used to highlight the instability of off-policy TD with function approximation, and (2) the Collision Problem, shown in Figure 2, used in prior work to highlight the advantages of ETD compared with gradient TD methods such as TDC (Ghiassian et al., 2018). In both cases we use linear function approximation, with a feature representation that includes significant generalization. In the appendix we also report experiments with Baird's counterexample (Baird, 1995).

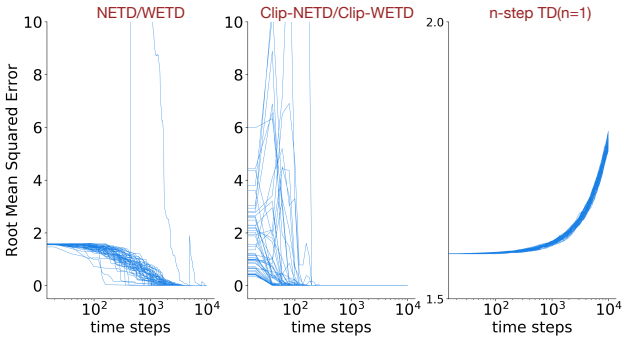
The results that follow are produced by extensive sweeps over the key hyper-parameters: we tested all combinations of the learning rate $\alpha \in \{2^i \mid i \in -14, -13, ..., -2\}$ and bootstrap length $n \in \{1, 2..., 5\}$; we selected the best hyper-parameters for each method by computing the RMSE over all time steps, and averaging results over many independent replications of the experiment—50 runs for the two-state MDP and 200 for the Collision problem. We then report both learning curves—plotting the RMSE over time for the best hyper-parameter setting from the sweep, and parameter studies—showing the average total RMSE for each algorithm, and for each combination of n, and α.

3.1. Two-state MDP

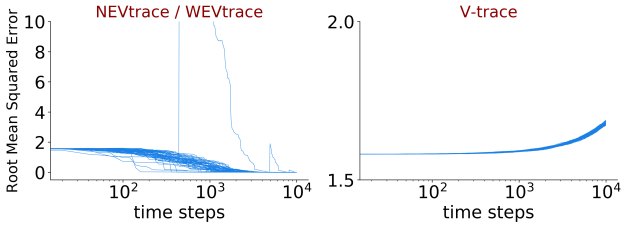
This two-state MDP, illustrated in Fig 1, is classical off-policy policy evaluation problem. Training data is generated by a random walk behavior policy. The task is to evaluate the target policy that always goes right, and ends up stuck in the second state forever. The values for the two states are approximated as θ and 2θ with state features being scalars 1 and 2. The discount γ is 0.9. Rewards are zero everywhere.

First we examine the convergence properties of various methods using the 1-step TD, a.k.a. TD(0) learning target. Empirically and theoretically, smaller bootstrap lengths n are more likely to induce divergence in learning (further analysis and empirical evidence is in the appendix). Notice that the emphatic traces NETD and WETD are equivalent when applied to TD(0) (compare the formula of NETD in Eq. 11 with that of WETD in Eq. 9 when $n = 1$).

Figure 4(a) presents Root Mean Squared Error (RMSE) over training time for NETD (WETD) and Clip-NETD (Clip-WETD), with the baseline learner TD(0) in the bottom panel. TD(0) diverged faster as training goes on. NETD converged slowly and exhibited significant instability: occasionally runs diverged even late into training. Clip-NETD by comparison learned quickly and exhibited low variance with no instability after some initial fluctuations. Since this small diagnostic MDP was designed to induce instability in learning, the initial fluctuations are expected due to the adversarial initialization of the value function parameters. Note we plot



(a) n-step TD target



(b) V-trace target

Figure 4. Stability and learning speed in the two-state problem (with $n = 1$ and 50 runs). Each plot reports the Root Mean Squared Error for all random seeds, using the best learning rate α for each method. (a) TD(0) slowly diverged; NETD learned slowly and exhibited significant instability, even late in learning. Clip-NETD learned quickly and exhibits no instability beyond a few initial fluctuations. (b) All runs of V-trace diverged, regardless of α; NEVtrace did converge, but exhibited instability in some runs. Note the log scale on x-axis.

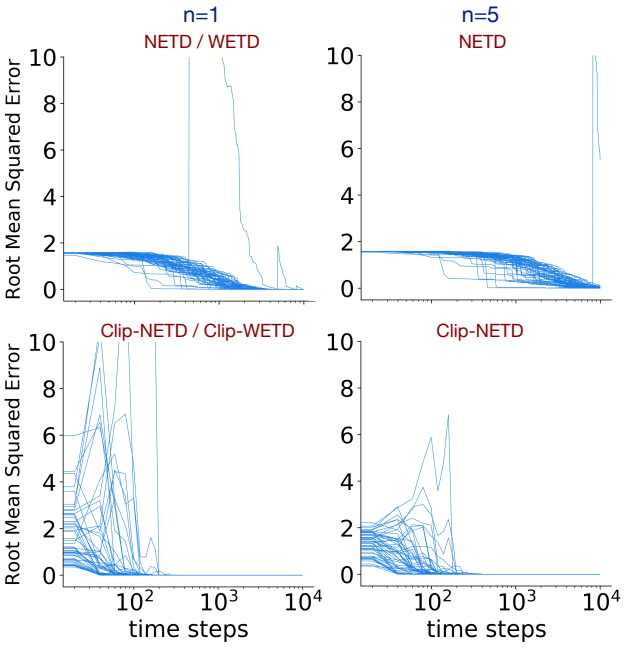


Figure 5. Variance reduction due to clipping the IS weights in emphatic trace computation. Left column: $n = 1$, right column: $n = 5$, both with 50 runs. Clip-NETD (Eq. 17) and Clip-WETD (Eq. 16) exhibit faster learning with no instability issues compared with the non-clipped version NETD / WETD. As n gets larger, NETD becomes more stable but still exhibits spikes late in learning, whereas Clip-NETD remains stable and improves in initial learning. Note the log scale on x-axis.

the error in the value estimates, but the algorithms are not directly optimizing value error. This is similar to previous results of Sutton & Barto (2018); Sutton et al. (2016), and like previous works, we plot all individual runs in order to highlight any instability in training. Overall, clipping IS weights proved to be an effective way of variance reduction at any bootstrap length n. Figure 5 shows an example of the variance reduction effect for both $n = 1$ (left column) and $n = 5$ (right column). In both cases, Clip-NETD learned faster with no instability issues.

Figure 4(b) compares NEVtrace (or WEVtrace) to the corresponding V-trace baseline at $n = 1$, that is equivalently, TD(0) with clipped IS weights. While V-trace diverged, NEVtrace converged slowly, although with instability issues and occasional spikes in error late in training in a subset of the runs.

The full set of experiment results on the two-state MDP for all algorithms listed in Table 1 are included in the appendix, as well as results on the Baird's MDP with $n = 1$ and $n = 5$, which yield similar conclusions.

3.2. Collision Problem

In the Collision Problem (illustrated in Fig. 2), states are aligned in a hallway and the agent can move forward or retreat. Episodes begin in one of the first four states, and terminates after 100 time steps. The reward is zero on every transition, except on the transition into the last state S_9. The behavior policy always moves forward in the starting states, and outside of this area, either moves forward or retreats to the starting states with equal probability, except in the last trapping state S_9. The target policy moves forward in every state. We examine the emphatic algorithms in this environment through a hyper-parameter sensitivity study on the learning rate α and bootstrap length n. Figure 3 presents the mean RMSE averaged across 200 runs of the emphatic algorithms and baselines n-step TD and V-trace at $n = 2$, varying the learning rate α. Emphatic algorithms achieved best performance—with best performance using smaller learning rates compared to the two baselines—consistent with previous results on ETD(λ) in this task (Ghiassian et al., 2018). Additional training curves and hyper-parameter study plots for $n = 1, 2, 3, 5$ are in the appendix, supporting the same conclusion.

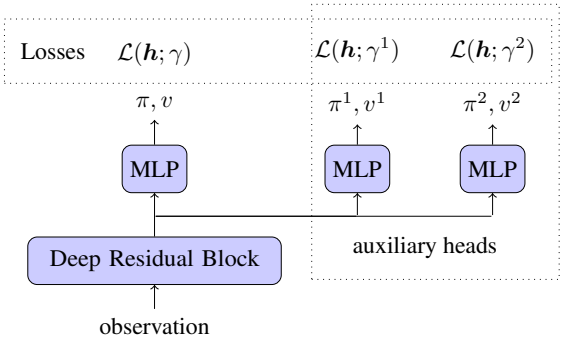


Figure 6. Block diagram of Surreal, with one main head and two auxiliary heads. IMPALA loss on each head uses different discounts $\gamma, \gamma^1, \gamma^2$. Let $\boldsymbol{h}$ denote the neural network model. The behavior policy is fixed to be π.

4. Experiments at Scale

Our ultimate goal is to design emphatic algorithms that improve off-policy learning at scale, especially on actor-critic agents. Thus we evaluated the emphatic algorithms on Atari games from the Arcade Learning Environment (Bellemare et al., 2013), a widely used deep RL benchmark.

Data We use the raw pixel observations in RGB as they are provided by the environment, without down sampling or gray scaling them. We also use an action repeat of 4, with max pooling over the last two frames and the life termination signal. This setup is similar to IMPALA (Espeholt et al., 2018) with the only difference being using the raw frames instead of down and gray scaled ones. In order to compare with closely related previous works, we adopted the conventional 200M frames training regime using online updates without experience replay.

Agent StacX (Zahavy et al., 2020) and UNREAL (Jaderberg et al., 2017) are both IMPALA-based agents that learn auxiliary tasks from experience generated by the main policy, in order to improve the shared representation. Inspired by their results, we investigated whether emphatic algorithms can help learn the auxiliary tasks better since they are learned off-policy, and in turn improve the agent performance. In particular, we used an IMPALA-based agent with two auxiliary heads, each head learning a different target policy for its own discount $\gamma, \gamma^1, \gamma^2$ (see Fig. 6 and the appendix for details on its network structures and hyperparameters). We call this agent *Surreal* as it fantasizes (learns off-policy) about two additional policies π^1, π^2 that discount the future rewards differently, without ever executing actions from them. We apply emphatic traces to the IMPALA learning updates on the two auxiliary heads. In order to reduce variance, we always clip the IS weights at 1 both in computing emphatic traces and in the V-trace target.

Algorithm 3 NETD-ACE Surreal.

Input: Bootstrapping length n**, discounts** $\gamma, \gamma^1, \gamma^2$**, number of actors** M**.**
Initialize Surreal neural network function $\boldsymbol{h}_0$,
Output initial policy for the main head π_0 from $\boldsymbol{h}_0$.
for *actor* $m \in [1, \dots, M]$ **do**
  Sample trajectory $(S_i^m, A_i^m, R_{i+1}^m)_{i=0}^{n-1} \sim \pi_0$.
  for *auxiliary head* $u = 1, 2$ **do**
    Initialize $F_0^{(n),m,u}, \dots, F_{n-1}^{(n),m,u} = 1$.
    for $i = 0, \dots, n-1$ **do**
      Set $\bar{\rho}_i^{m,u} = \min(1, \frac{\pi_0^u(A_i^m|S_i^m)}{\pi_0(A_i^m|S_i^m)})$.
    end
  end
end
for *timestep* $t \in [0, \dots, T]$ **do**
  Output main head policy π_t from neural network $\boldsymbol{h}_t$.
  for *actor* $m \in [1, \dots, M]$ **do**
    Sample $S_{t+n}^m, A_{t+n}^m, R_{t+n+1}^m \sim \pi_t$.
    Compute the main head IMPALA loss $\mathcal{L}_t^m(\boldsymbol{h};\gamma)$.
    for *auxiliary head* $u = 1, 2$ **do**
      Set $\bar{\rho}_{t+n}^{m,u} = \min(1, \frac{\pi_t^u(A_{t+n}^m|S_{t+n}^m)}{\pi_t(A_{t+n}^m|S_{t+n}^m)})$.
      if $t \geq n$ **then**
      $F_t^{(n),m,u} = \prod_{i=1}^n (\gamma_{t-i+1}^u \bar{\rho}_{t-i}^{m,u}) F_{t-n}^{(n),m,u} + 1$,
      end
      Weight the sum of IMPALA value and policy losses, plus IMPALA entropy loss:
      $\mathcal{E}_t^{m,u} = F_t^{(n),m,u} \mathcal{L}_t^{m,u}(\boldsymbol{h};\gamma^u) + \mathcal{H}_t^{m,u}$.
    end
  end
  Update neural network $\boldsymbol{h}_{t+1}$ using an average loss:
  $\mathcal{L}_t = \frac{1}{3M} \sum_m (\mathcal{L}_t^m + \mathcal{E}_t^{m,1} + \mathcal{E}_t^{m,2})$.
end
Return: h_T.

In a distributed system, we keep track of an emphatic trace for each actor's trajectories and aggregate the updates in a batch average at every time step. Algo. 3 outlines the pseudo-code for NETD-ACE Surreal as an example. For the implementation of Surreal, we used Jax libraries (Budden et al., 2020; Hennigan et al., 2020; Hessel et al., 2020) on a TPU Pod infrastructure called Sebulba (Hessel et al., 2021).

Evaluation We compute the median human normalized scores across 57 games, averaged over seeds and an *evaluation phase* without learning. To compare any two agents, we view their scores on 57 games as 57 independent pairs of samples, similar to how one would test significance of a medical treatment on a population of different people, rather than testing same treatment on the same person multiple times. The p-value is the probability of the null hypothesis

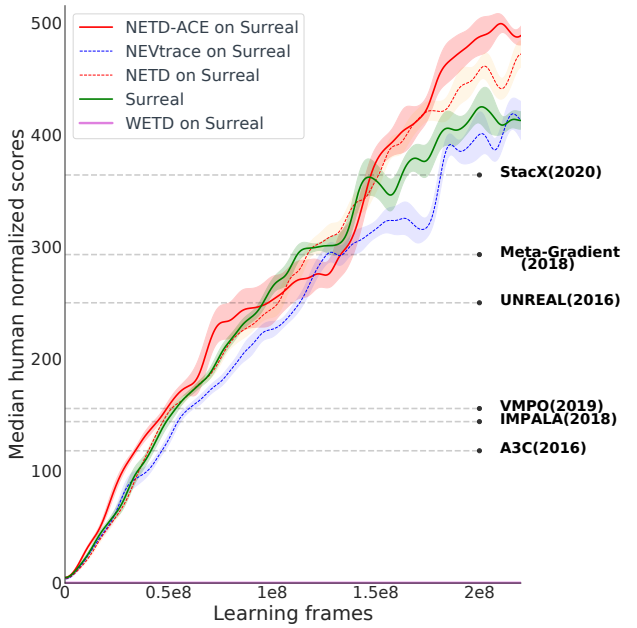


Figure 7. Learning curves of baseline Surreal and the emphatic traces (NETD, NETD-ACE, NEVtrace, WETD) applied to Surreal, in the fixed update scheme with $n = 10$, IS weights clipped to 1. Median human normalized scores are averaged across 3 random seeds with shaded areas denoting standard derivations.

that the algorithm performs equally using the *sign test* (Arbuthnot, 1712). Results might be thought of as statistically significant when $p < 0.05$.

Baselines Prior to applying emphatic traces, we found the best hyper-parameters for Surreal in the mixed and the fixed update schemes separately as our baselines. In the mixed update scheme, $n = 40, \alpha = 6 \cdot 10^{-4}$, max gradient norm $= 0.3$ yielded the best results. In the fixed update scheme, the best hyper-parameters for Surreal were $n = 10, \alpha = 2 \cdot 10^{-4}$, max gradient norm $= 1$.

Emphatic Results Since the emphatic traces are derived using the steady state distributions following fixed policies, we expect that they would impact the results more towards the end of learning, as the agent stabilizes its learned policy with learning rate decay. Empirically we observed the differences between algorithms start to show around 130M frames or 65% of learning frames.

In the mixed update scheme, we tested emphatic trace family WETD and its variants WETD-ACE, WEVtrace applied to the Surreal baseline. In order to reduce instability, we experimented with several variance reduction techniques, including: 1) ETD(λ, β), 2) interpolation, and 3) clipping. First, previous work on ETD(λ, β)(Hallak et al., 2016) suggested using a hyper-parameter β to replace the discount variable γ_t in the follow-on trace, which we applied to WETD. Second, we introduced a constant hyper-parameter $\eta \in (0, 1)$ such that $M_t^w = 1 - \eta(1 - \lambda_t) + \eta(1 - \lambda_t)F_t$, allowing us to modify the interpolation between a potentially large F_t and 1 to restrain the blow-up. Third, we clipped the values of ρ_{t-1} and/or directly clipped the values of F_t. However, despite of these efforts, WETD still diverged with exploding gradients (see WETD in Fig. 7).

Statistics	NETD-ACE	NETD	NEVtrace	Surreal
Median	**497.21**	427.69	317.50	403.47
Mean	**1793.47**	1507.85	1502.84	1565.42
40th percentile	**300.26**	268.43	186.95	258.35
30th percentile	162.91	**169.42**	118.27	163.90
20th percentile	**74.47**	70.74	28.28	65.60
10th percentile	4.88	4.3	4.91	4.25
# games >human	**45**/57	**45**/57	43/57	43/57

Table 2. Performance statistics for baseline Surreal and emphatic traces applied to Surreal in the fixed update scheme with $n = 10$, on 57 Atari games. Scores are human normalized, averaged across 3 random seeds and across the evaluation phase.

Next, we evaluated emphatic trace family NETD (dashed orange) and its variants NETD-ACE (solid red), NEVtrace (dashed blue), applied to Surreal, along with the baseline Surreal agent (solid green) using a fixed update scheme. The best Surreal baseline from our sweeps already surpassed the StacX scores (Zahavy et al., 2020). The results are averaged over 3 random seeds, and Fig. 7 depicts the learning curves and Table 2 summarizes all performance statistics. In particular, the best performing emphatic actor-critic agent NETD-ACE improved the median human normalized score from the baseline performance of 403% to 497%, the highest score for an RL agent without experience replay in the 200M frames data regime. It improved performance on 100 out of 57×3 Atari games compared to the baseline, with a p-value of 0.016, achieving 95% statistical significance.

5. Discussion

New emphatic algorithm families of WETD and NETD variants showed nice qualitative properties on off-policy diagnostic MDPs. For both families, clipping IS weights in computing emphatic traces turns out to be an effective way to reduce variance, so we applied this learning at scale. On Atari, we proposed a baseline agent Surreal that achieved a strong median human normalized score 403%, and is suitable for testing off-policy learning on auxiliary controls. The WETD family were unstable at scale, whereas the NETD family performed well, particularly the emphatic actor-critic agent NETD-ACE. For future work, we would like to investigate applying emphatic traces to a variety of off-policy learning targets and settings at scale.

References

Arbuthnot, J. II. An argument for divine providence, taken from the constant regularity observ'd in the births of both sexes. By Dr. John Arbuthnott, Physitian in Ordinary to Her Majesty, and Fellow of the College of Physitians and the Royal Society. *Philosophical Transactions of the Royal Society of London*, 27(328):186–190, 1712.

Baird, L. Residual algorithms: Reinforcement learning with function approximation. *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 30–37, 1995.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253279, 2013.

Bellman, R. A markovian decision process. *Journal of Mathematics and Mechanics*, 1957.

Budden, D., Hessel, M., Quan, J., Kapturowski, S., Baumli, K., Bhupatiraju, S., Guy, A., and King, M. RLax: Reinforcement Learning in JAX, 2020. URL http://github.com/deepmind/rlax.

Degris, T. and Modayil, J. Scaling-up knowledge for a cognizant robot. *AAAI Spring Symposium: Designing Intelligent Robots*, 2012.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. *CoRR*, 2018.

Ghiassian, S., Patterson, A., White, M., Sutton, R. S., and White, A. Online off-policy prediction. *arXiv preprint arXiv:1811.02597*, 2018.

Hallak, A., Tamar, A., Munos, R., and Mannor, S. Generalized emphatic temporal difference learning: Bias-variance analysis. *In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016.

Hennigan, T., Cai, T., Norman, T., and Babuschkin, I. Haiku: Sonnet for JAX, 2020. URL http://github.com/deepmind/dm-haiku.

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and van Hasselt, H. Multi-task deep reinforcement learning with popart. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3796–3803, 2019.

Hessel, M., Budden, D., Viola, F., Rosca, M., Sezener, E., and Hennigan, T. Optax: composable gradient transformation and optimisation, in JAX!, 2020. URL http://github.com/deepmind/optax.

Hessel, M., Kroiss, M., Clark, A., Kemaev, I., Quan, J., Keck, T., Viola, F., and van Hasselt, H. Podracer architectures for scalable reinforcement learning. 2021. URL https://arxiv.org/pdf/2104.06272.pdf.

Imani, E., Graves, E., and White, M. An off-policy policy gradient theorem using emphatic weightings. *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

Jaderberg, M., Mnih, V., Czarnecki, W., Schaul, T., Leibo, J., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. *ICLR*, 2017.

Lin, L. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992.

Mahmood, A. R., Yu, H., and Sutton, R. S. Multi-step off-policy learning without importance sampling ratios. *arXiv preprint arXiv:1702.03006*, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 2015.

Precup, D., Sutton, R. S., and Dasgupta, S. Off-policy temporal-difference learning with function approximation. *ICML*, pp. 417–424, 2001.

Sutton, R. S. Implementation details of the td(λ) procedure for the case of vector predictions and backpropagation. *GTE Laboratories Technical Note TN87-509.1*, 1987.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 2018.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems 13*, 12:1057–1063, 2000.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. Fast gradient-descent methods for temporal-difference learning with linear function approximation. pp. 993–1000. ACM, 2009.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768, 2011.

Sutton, R. S., Mahmood, A. R., and White, M. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 17 (1):2603–2631, 2016.

Tsitsiklis, J. N. and Van Roy, B. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. *CoRR*, abs/1812.02648, 2018. URL http://arxiv.org/abs/1812.02648.

van Hasselt, H., Madjiheurem, S., Hessel, M., Silver, D., Barreto, A., and Borsa, D. Expected eligibility traces. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):9997–10005, May 2021.

Watkins, C. J. C. H. Learning from delayed rewards. 1989.

Yu, H. On convergence of emphatic temporal-difference learning. *JMLR: Workshop and Conference Proceedings*, 40:128, 2015.

Zahavy, T., Xu, Z., Veeriah, V., Hessel, M., Oh, J., van Hasselt, H., Silver, D., and Singh, S. A self-tuning actor-critic algorithm. *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.