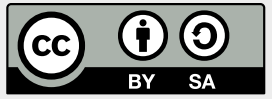
Eligibility Traces, Impala, R2D2, Agent57

Milan Straka

November 28, 2022

EUROPEAN UNION
European Structural and Investment Fund
Operational Programme Research,
Development and Education

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics



Off-policy Correction Using Control Variates

Denoting the TD error as $\delta_t \stackrel{\text{def}}{=} R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$, we can write the n-step estimated return as a sum of TD errors:

$$G_{t:t+n} = V(S_t) + \sum_{i=0}^{n-1} \gamma^i \delta_{t+i}.$$

Furthermore, denoting the importance sampling ratio $\rho_t \stackrel{\text{def}}{=} \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$, $\quad \rho_{t:t+n} \stackrel{\text{def}}{=} \prod_{i=0}^{n} \rho_{t+i}$, we can introduce the **control variate** to the estimate

$$G^{\text{CV}}_{t:t+n} \stackrel{\text{def}}{=} \rho_t \left(R_{t+1} + \gamma G^{\text{CV}}_{t+1:t+n}\right) + (1 - \rho_t) V(S_t),$$

which can then be written as

$$G^{\text{CV}}_{t:t+n} = V(S_t) + \sum_{i=0}^{n-1} \gamma^i \rho_{t:t+i} \delta_{t+i}.$$

Eligibility Traces

Eligibility traces are a mechanism of combining multiple n-step return estimates for various values of n.

First note instead of an n-step return, we can use any average of n-step returns for different values of n, for example $\frac{2}{3}G_{t:t+2} + \frac{1}{3}G_{t:t+4}$.

λ-return

For a given $\lambda \in [0, 1]$, we define λ-**return** as

$$G_t^\lambda \stackrel{\text{def}}{=} (1-\lambda) \sum_{i=1}^{\infty} \lambda^{i-1} G_{t:t+i}.$$

Alternatively, the λ-return can be written recursively as

$$G_t^\lambda = (1-\lambda) G_{t:t+1} + \lambda (R_{t+1} + \gamma G_{t+1}^\lambda).$$

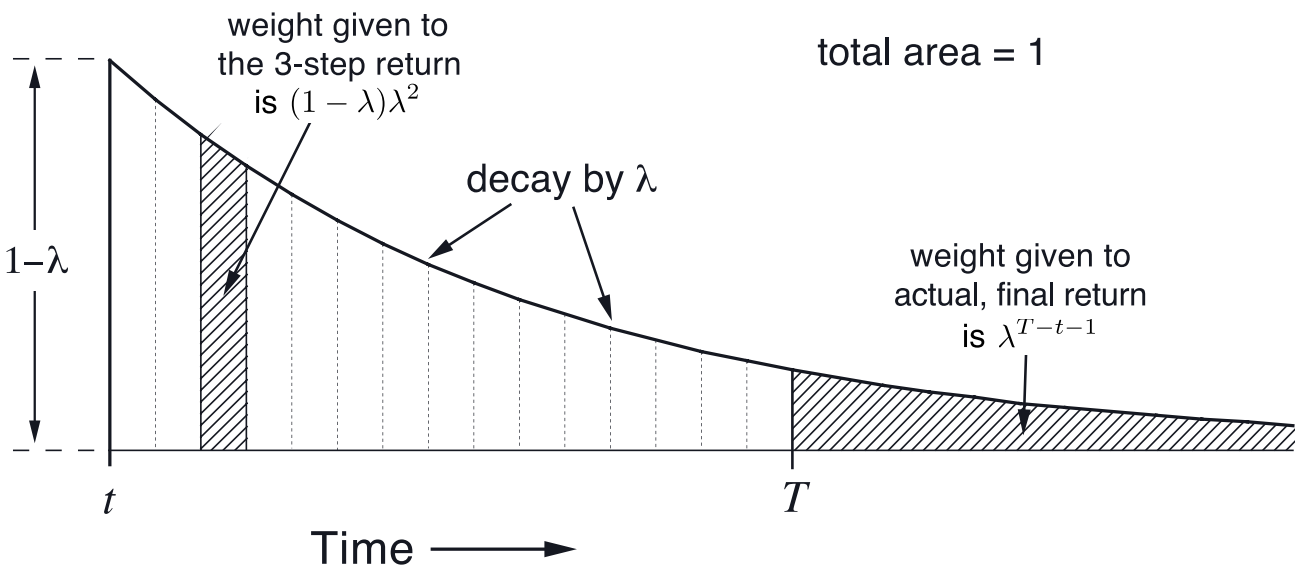


Figure 12.2: Weighting given in the λ-return to each of the n-step returns.
Figure 12.2 of "Reinforcement Learning: An Introduction, Second Edition".

In an episodic task with time of termination T, we can rewrite the λ-return to

$$G_t^\lambda = (1-\lambda) \sum_{i=1}^{T-t-1} \lambda^{i-1} G_{t:t+i} + \lambda^{T-t-1} G_t.$$

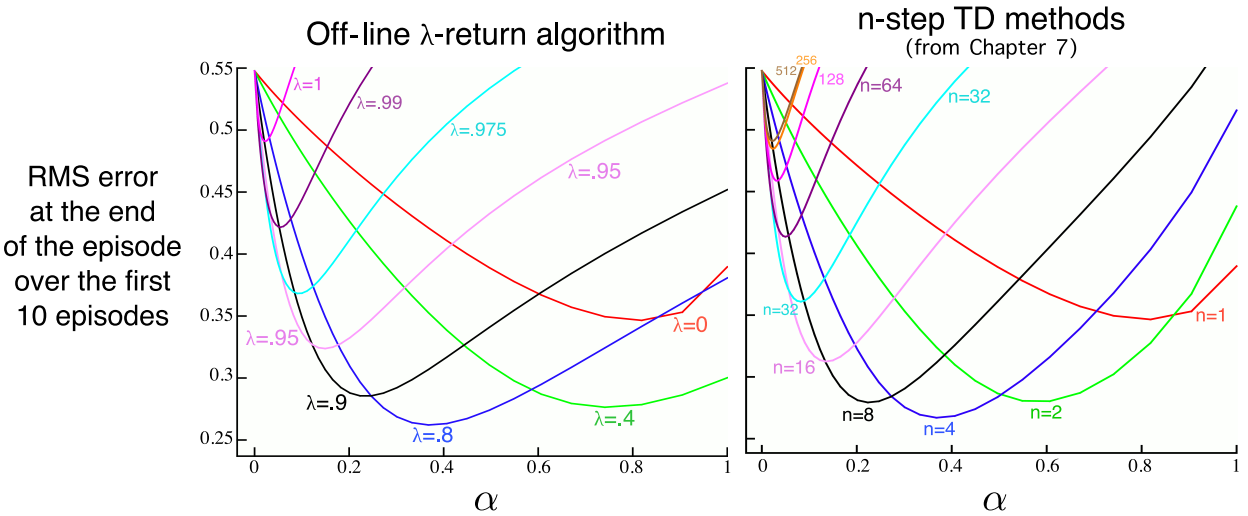


Figure 12.3: 19-state Random walk results (Example 7.1): Performance of the off-line λ-return algorithm alongside that of the n-step TD methods. In both case, intermediate values of the bootstrapping parameter (λ or n) performed best. The results with the off-line λ-return algorithm are slightly better at the best values of α and λ, and at high α.

Figure 12.3 of "Reinforcement Learning: An Introduction, Second Edition".

Truncated λ-return

We might also set a limit on the largest value of n, obtaining **truncated λ-return**

$$G^{\lambda}_{t:t+n} \stackrel{\text{def}}{=} (1-\lambda)\sum_{i=1}^{n-1}\lambda^{i-1}G_{t:t+i} + \lambda^{n-1}G_{t:t+n}.$$

The truncated λ return can be again written recursively as

$$G^{\lambda}_{t:t+n} = (1-\lambda)G_{t:t+1} + \lambda(R_{t+1} + \gamma G^{\lambda}_{t+1:t+n}), \;\; G^{\lambda}_{t:t+1} = G_{t:t+1}.$$

Similarly to before, we can express the truncated λ return as a sum of TD errors

$$\begin{aligned} G^{\lambda}_{t:t+n} - V(S_t) &= (1-\lambda)\big(R_{t+1} + \gamma V(S_{t+1})\big) + \lambda(R_{t+1} + \gamma G^{\lambda}_{t+1:t+n}) - V(S_t) \\ &= R_{t+1} + \gamma V(S_{t+1}) - V(S_t) + \lambda\gamma\big(G^{\lambda}_{t+1:t+n} - V(S_{t+1})\big), \end{aligned}$$

obtaining an analogous estimate $G^{\lambda}_{t:t+n} = V(S_t) + \sum_{i=0}^{n-1}\gamma^i\lambda^i\delta_{t+i}$.

Variable λs

The (truncated) λ-return can be generalized to utilize different λ_i at each step i. Notably, we can generalize the recursive definition

$$G^{\lambda}_{t:t+n} = (1-\lambda)G_{t:t+1} + \lambda(R_{t+1} + \gamma G^{\lambda}_{t+1:t+n})$$

to

$$G^{\lambda_i}_{t:t+n} = (1-\lambda_{t+1})G_{t:t+1} + \lambda_{t+1}(R_{t+1} + \gamma G^{\lambda_i}_{t+1:t+n}),$$

and express this quantity again by a sum of TD errors:

$$G^{\lambda_i}_{t:t+n} = V(S_t) + \sum_{i=0}^{n-1} \gamma^i \left(\prod_{j=1}^{i} \lambda_{t+j}\right) \delta_{t+i}.$$

Off-policy Traces with Control Variates

Finally, we can combine the eligibility traces with off-policy estimation using control variates:

$$G^{\lambda,\mathrm{CV}}_{t:t+n} \overset{\text{def}}{=} (1-\lambda)\sum_{i=1}^{n-1}\lambda^{i-1}G^{\mathrm{CV}}_{t:t+i} + \lambda^{n-1}G^{\mathrm{CV}}_{t:t+n}.$$

Recalling that

$$G^{\mathrm{CV}}_{t:t+n} = \rho_t\big(R_{t+1} + \gamma G^{\mathrm{CV}}_{t+1:t+n}\big) + (1-\rho_t)V(S_t),$$

we can rewrite $G^{\lambda,\mathrm{CV}}_{t:t+n}$ recursively as

$$G^{\lambda,\mathrm{CV}}_{t:t+n} = (1-\lambda)G^{\mathrm{CV}}_{t:t+1} + \lambda\Big(\rho_t\big(R_{t+1} + \gamma G^{\lambda,\mathrm{CV}}_{t+1:t+n}\big) + (1-\rho_t)V(S_t)\Big),$$

which we can simplify by expanding $G^{\mathrm{CV}}_{t:t+1} = \rho_t(R_{t+1} + \gamma V(S_{t+1})) + (1-\rho_t)V(S_t)$ to

$$G^{\lambda,\mathrm{CV}}_{t:t+n} - V(S_t) = \rho_t\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big) + \gamma\lambda\rho_t\big(G^{\lambda,\mathrm{CV}}_{t+1:t+n} - V(S_{t+1})\big).$$

Off-policy Traces with Control Variates

Consequently, analogously as before, we can write the off-policy traces estimate with control variates as

$$G^{\lambda,\mathrm{CV}}_{t:t+n} = V(S_t) + \sum_{i=0}^{n-1} \gamma^i \lambda^i \rho_{t:t+i} \delta_{t+i},$$

and by repeating the above derivation we can extend the result also for time-variable λ_i, we obtain

$$G^{\lambda_\mathrm{i},\mathrm{CV}}_{t:t+n} = V(S_t) + \sum_{i=0}^{n-1} \gamma^i \left(\prod_{j=1}^{i} \lambda_{t+j} \right) \rho_{t:t+i} \delta_{t+i}.$$

Return Recapitulation

Recursive definition	Formulation with TD errors
$G_{t:t+n} \overset{\text{def}}{=} R_{t+1} + \gamma G_{t+1:t+n}$	$V(S_t) + \sum_{i=0}^{n-1} \gamma^i \delta_{t+i}$
$G^{\text{IS}}_{t:t+n} \overset{\text{def}}{=} \rho_t \left(R_{t+1} + \gamma G^{\text{IS}}_{t+1:t+n}\right)$	
$G^{\text{CV}}_{t:t+n} \overset{\text{def}}{=} \rho_t \left(R_{t+1} + \gamma G^{\text{CV}}_{t+1:t+n}\right) + (1-\rho_t) V(S_t)$	$V(S_t) + \sum_{i=0}^{n-1} \gamma^i \rho_{t:t+i} \delta_{t+i}$
$G^{\lambda}_{t:t+n} \overset{\text{def}}{=} (1-\lambda) G_{t:t+1} + \lambda \left(R_{t+1} + \gamma G^{\lambda}_{t+1:t+n}\right)$	$V(S_t) + \sum_{i=0}^{n-1} \gamma^i \lambda^i \delta_{t+i}$
$G^{\lambda_i}_{t:t+n} \overset{\text{def}}{=} (1-\lambda_{t+1}) G_{t:t+1} + \lambda_{t+1} \left(R_{t+1} + \gamma G^{\lambda_i}_{t+1:t+n}\right)$	$V(S_t) + \sum_{i=0}^{n-1} \gamma^i \left(\prod_{j=1}^{i} \lambda_{t+j}\right) \delta_{t+i}$
$G^{\lambda,\text{CV}}_{t:t+n} \overset{\text{def}}{=} (1-\lambda) G^{\text{CV}}_{t:t+1} + \lambda \left(\rho_t \left(R_{t+1} + \gamma G^{\lambda,\text{CV}}_{t+1:t+n}\right) + (1-\rho_t) V(S_t)\right)$	$V(S_t) + \sum_{i=0}^{n-1} \gamma^i \lambda^i \rho_{t:t+i} \delta_{t+i}$
$G^{\lambda_i,\text{CV}}_{t:t+n} \overset{\text{def}}{=} (1-\lambda_{t+1}) G^{\text{CV}}_{t:t+1} + \lambda_{t+1} \left(\rho_t \left(R_{t+1} + \gamma G^{\lambda_i,\text{CV}}_{t+1:t+n}\right) + (1-\rho_t) V(S_t)\right)$	$V(S_t) + \sum_{i=0}^{n-1} \gamma^i \left(\prod_{j=1}^{i} \lambda_{t+j}\right) \rho_{t:t+i} \delta_{t+i}$

We have defined the λ-return in the so-called **forward view**.

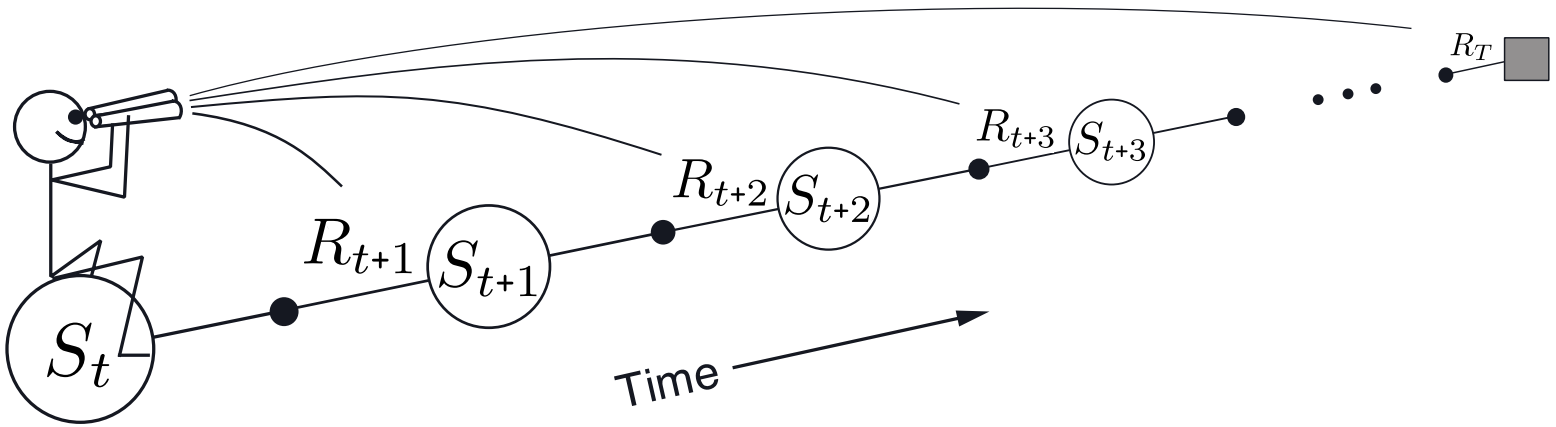


Figure 12.4: The forward view. We decide how to update each state by looking forward to future rewards and states.

Figure 12.4 of "Reinforcement Learning: An Introduction, Second Edition".

However, to allow on-line updates, we might consider also the **backward view**

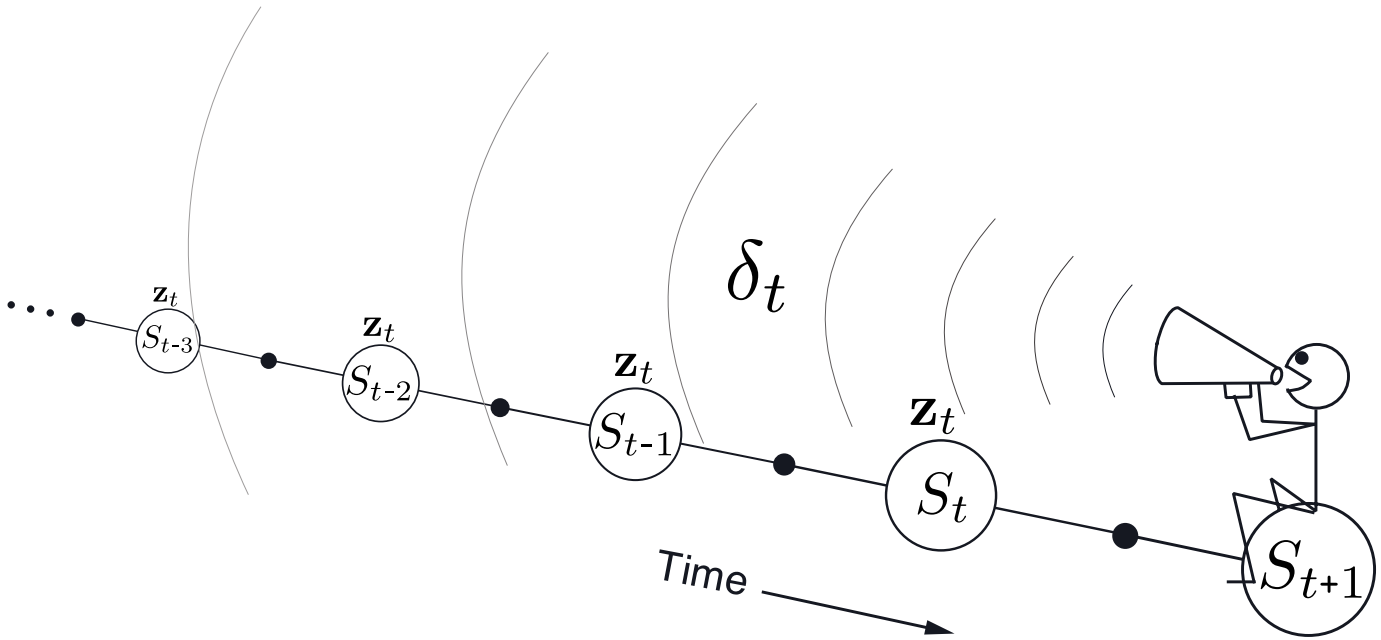


Figure 12.5: The backward or mechanistic view of TD(λ). Each update depends on the current TD error combined with the current eligibility traces of past events.

Figure 12.5 of "Reinforcement Learning: An Introduction, Second Edition".

TD(λ) is an algorithm implementing on-line policy evaluation utilizing the backward view.

Semi-gradient TD(λ) for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameters: step size $\alpha > 0$, trace decay rate $\lambda \in [0, 1]$
Initialize value-function weights $\mathbf{w}$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  Initialize S
  $\mathbf{z} \leftarrow \mathbf{0}$ (a d-dimensional vector)
  Loop for each step of episode:
  | Choose $A \sim \pi(\cdot|S)$
  | Take action A, observe R, S'
  | $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z} + \nabla\hat{v}(S,\mathbf{w})$
  | $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$
  | $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{z}$
  | $S \leftarrow S'$
  until S' is terminal

Algorithm 12.2 of "Reinforcement Learning: An Introduction, Second Edition".

V-trace

V-trace is a modified version of n-step return with off-policy correction, defined in the Feb 2018 IMPALA paper as (using the notation from the paper):

$$G_{t:t+n}^{\text{V-trace}} \stackrel{\text{def}}{=} V(S_t) + \sum_{i=0}^{n-1} \gamma^i \left(\prod_{j=0}^{i-1} \bar{c}_{t+j} \right) \bar{\rho}_{t+i} \delta_{t+i},$$

where $\bar{\rho}_t$ and $\bar{c}_t$ are the truncated importance sampling ratios for $\bar{\rho} \geq \bar{c}$:

$$\bar{\rho}_t \stackrel{\text{def}}{=} \min\left(\bar{\rho}, \frac{\pi(A_t|S_t)}{b(A_t|S_t)}\right), \quad \bar{c}_t \stackrel{\text{def}}{=} \min\left(\bar{c}, \frac{\pi(A_t|S_t)}{b(A_t|S_t)}\right).$$

Note that if $b = \pi$ and assuming $\bar{c} \geq 1$, v_s reduces to n-step Bellman target.

V-trace

Note that the truncated IS weights $\bar{\rho}_t$ and $\bar{c}_t$ play different roles:

- The $\bar{\rho}_t$ appears defines the fixed point of the update rule. For $\bar{\rho} = \infty$, the target is the value function v_π, if $\bar{\rho} < \infty$, the fixed point is somewhere between v_π and v_b. Notice that we do not compute a product of these $\bar{\rho}_t$ coefficients.

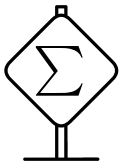
  Concretely, the fixed point of an operator defined by $G_{t:t+n}^{\text{V-trace}}$ corresponds to a value function of the policy

$$\pi_{\bar{\rho}}(a|s) \propto \min\left(\bar{\rho} b(a|s), \pi(a|s)\right).$$

- The $\bar{c}_t$ impacts the speed of convergence (the contraction rate of the Bellman operator), not the sought policy. Because a product of the $\bar{c}_t$ ratios is computed, it plays an important role in variance reduction.

However, the paper utilizes $\bar{c} = 1$ and out of $\bar{\rho} \in \{1, 10, 100\}$, $\bar{\rho} = 1$ works empirically the best, so the distinction between $\bar{c}_t$ and $\bar{\rho}_t$ is not useful in practice.

V-trace Analysis



Let us define the (untruncated for simplicity; similar results can be proven for a truncated one) V-trace operator $\mathcal{R}$ as:

$$\mathcal{R}V(S_t) \stackrel{\text{def}}{=} V(S_t) + \mathbb{E}_b\left[\sum\nolimits_{i\geq 0} \gamma^i \left(\prod\nolimits_{j=0}^{i-1} \bar{c}_{t+j}\right) \bar{\rho}_{t+i}\delta_{t+i}\right],$$

where the expectation $\mathbb{E}_b$ is with respect to trajectories generated by behaviour policy b.

Assuming there exists $\beta \in (0, 1]$ such that $\mathbb{E}_b\bar{\rho}_0 \geq \beta$, it can be proven (see Theorem 1 in Appendix A.1 in the Impala paper if interested) that such an operator is a contraction with a contraction constant

$$\gamma^{-1} - (\gamma^{-1} - 1)\underbrace{\sum\nolimits_{i\geq 0} \gamma^i \mathbb{E}_b\left[\left(\prod\nolimits_{j=0}^{i-1} \bar{c}_j\right) \bar{\rho}_i\right]}_{\geq 1+\gamma\mathbb{E}_b\bar{\rho}_0} \leq 1 - (1-\gamma)\beta < 1,$$

therefore, $\mathcal{R}$ has a unique fixed point.

We now prove that the fixed point of $\mathcal{R}$ is $V^{\pi_{\bar{\rho}}}$. We have:

$$\begin{aligned}
\mathbb{E}_b\left[\bar{\rho}_t\delta_t\right] &= \mathbb{E}_b\left[\bar{\rho}_t\left(R_{t+1}+\gamma V^{\pi_{\bar{\rho}}}(S_{t+1})-V^{\pi_{\bar{\rho}}}(S_t)\right)\middle|S_t\right] \\
&= \sum_a b(a|S_t)\min\left(\bar{\rho},\frac{\pi(a|S_t)}{b(a|S_t)}\right)\left[R_{t+1}+\gamma\mathbb{E}_{s'\sim p(S_t,a)}V^{\pi_{\bar{\rho}}}(s')-V^{\pi_{\bar{\rho}}}(S_t)\right] \\
&= \underbrace{\sum_a \pi_{\bar{\rho}}(a|S_t)\left[R_{t+1}+\gamma\mathbb{E}_{s'\sim p(S_t,a)}V^{\pi_{\bar{\rho}}}(s')-V^{\pi_{\bar{\rho}}}(S_t)\right]}_{=0}\sum_{a'}\min\left(\bar{\rho}b(a'|S_t),\pi(a'|S_t)\right) \\
&= 0,
\end{aligned}$$

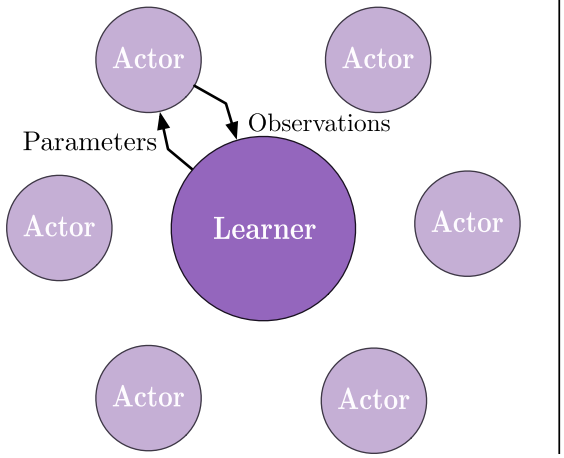
where the tagged part is zero, since it is the Bellman equation for $V^{\pi_{\bar{\rho}}}$. This shows that $\mathcal{R}V^{\pi_{\bar{\rho}}}(s) = V^{\pi_{\bar{\rho}}}(s) + \mathbb{E}_b\left[\sum_{i\geq 0}\gamma^i\left(\prod_{j=0}^{i-1}\bar{c}_{t+j}\right)\bar{\rho}_{t+i}\delta_{t+i}\right] = V^{\pi_{\bar{\rho}}}$, and therefore $V^{\pi_{\bar{\rho}}}$ is the unique fixed point of $\mathcal{R}$.

Consequently, in $G_{t:t+n}^{\lambda_i,\mathrm{CV}} = V(S_t) + \sum_{i=0}^{n-1}\gamma^i\left(\prod_{j=1}^{i}\lambda_{t+j}\right)\rho_{t:t+i}\delta_{t+i}$, only the last ρ_{t+i} from every $\rho_{t:t+i}$ is actually needed for off-policy correction; $\rho_{t:t+i-1}$ can be considered as traces.

IMPALA

Impala (**Imp**ortance Weighted **A**ctor-**L**earner **A**rchitecture) was suggested in Feb 2018 paper and allows massively distributed implementation of an actor-critic-like learning algorithm.

Compared to A3C-based agents, which communicate gradients with respect to the parameters of the policy, IMPALA actors communicate trajectories to the centralized learner.



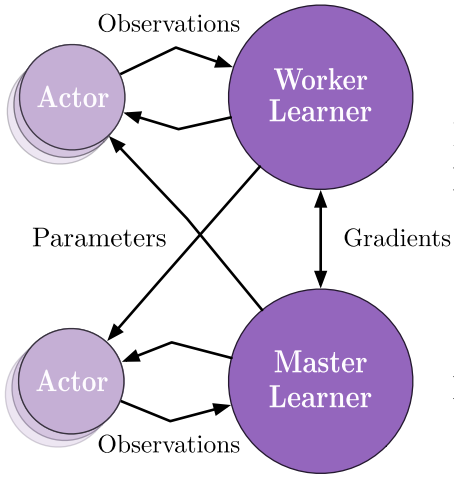


Figure 1 of "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures" by Lasse Espeholt et al.

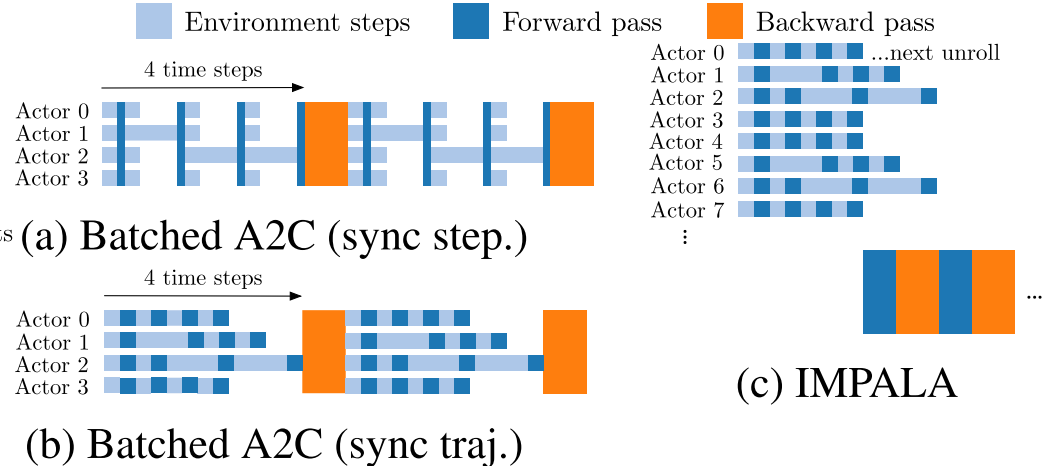


Figure 2 of "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures" by Lasse Espeholt et al.

If many actors are used, the policy used to generate a trajectory can lag behind the latest policy. Therefore, the V-trace off-policy actor-critic algorithm is employed.

Consider a parametrized functions computing $v(s;\boldsymbol{\theta})$ and $\pi(a|s;\boldsymbol{\omega})$, we update the critic in the direction of

$$\Big(G^{\text{V-trace}}_{t:t+n} - v(S_t;\boldsymbol{\theta})\Big)\nabla_{\boldsymbol{\theta}} v(S_t;\boldsymbol{\theta}),$$

and the actor in the direction of the policy gradient

$$\bar{\rho}_t \nabla_{\boldsymbol{\omega}} \log \pi(A_t|S_t;\boldsymbol{\omega})\big(R_{t+1} + \gamma G^{\text{V-trace}}_{t+1:t+n} - v(S_t;\boldsymbol{\theta})\big).$$

Finally, we again add the entropy regularization term $\beta H\big(\pi(\cdot|S_t;\boldsymbol{\omega})\big)$ to the loss function.

Architecture	CPUs	GPUs[1]	FPS[2]	
Single-Machine			Task 1	Task 2
A3C 32 workers	64	0	6.5K	9K
Batched A2C (sync step)	48	0	9K	5K
Batched A2C (sync step)	48	1	13K	5.5K
Batched A2C (sync traj.)	48	0	16K	17.5K
Batched A2C (dyn. batch)	48	1	16K	13K
IMPALA 48 actors	48	0	17K	20.5K
IMPALA (dyn. batch) 48 actors[3]	48	1	21K	24K
Distributed				
A3C	200	0	46K	50K
IMPALA	150	1	80K	
IMPALA (optimised)	375	1	200K	
IMPALA (optimised) batch 128	500	1	250K	

[1] Nvidia P100 [2] In frames/sec (4 times the agent steps due to action repeat). [3] Limited by amount of rendering possible on a single machine.

Table 1 of "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures" by Lasse Espeholt et al.

IMPALA – Population Based Training

For Atari experiments, population based training with a population of 24 agents is used to adapt entropy regularization, learning rate, RMSProp ε and the global gradient norm clipping threshold.

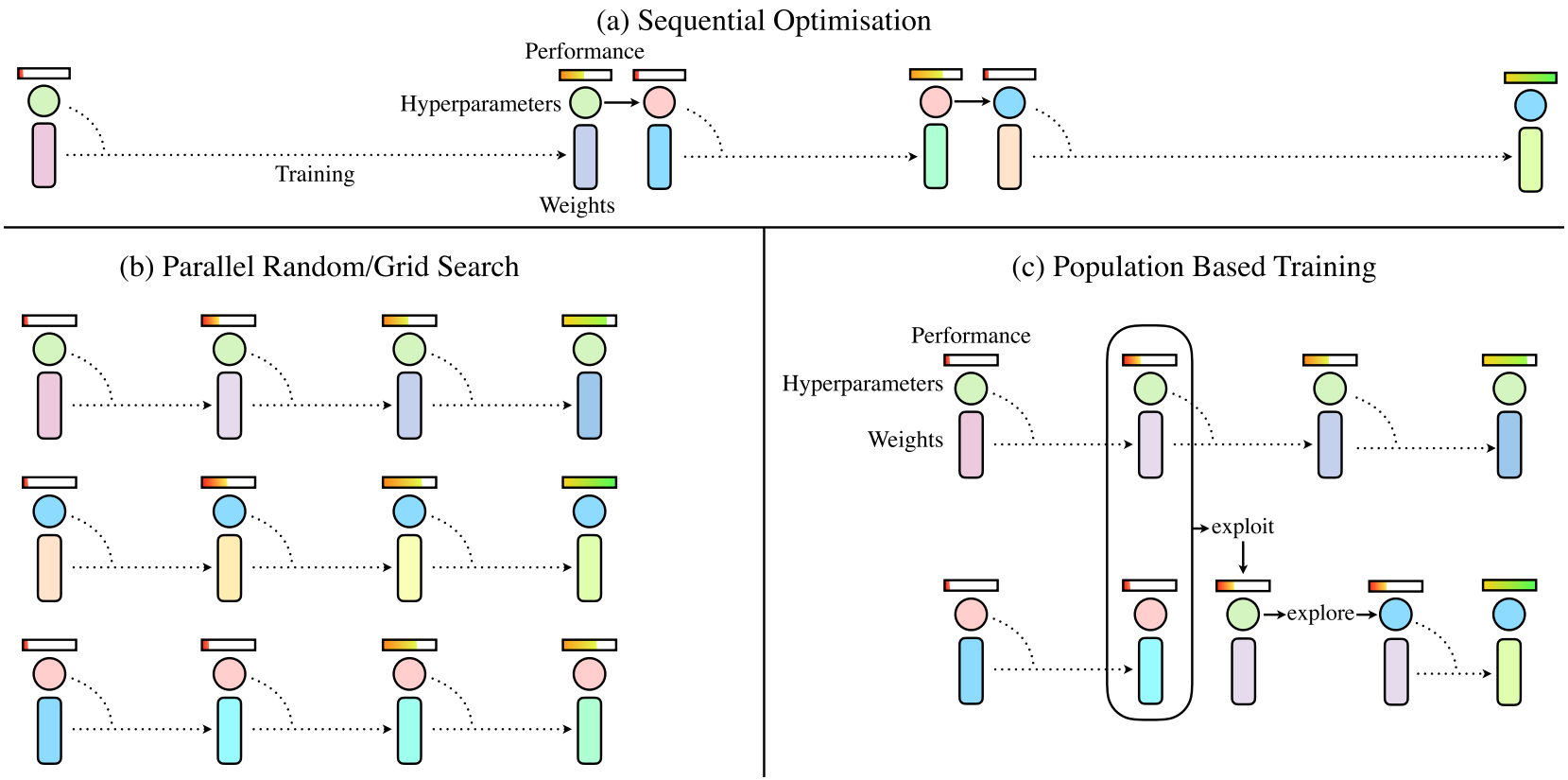


Figure 1 of "Population Based Training of Neural Networks" by Max Jaderberg et al.

IMPALA – Population Based Training

For Atari experiments, population based training with a population of 24 agents is used to adapt entropy regularization, learning rate, RMSProp ε and the global gradient norm clipping threshold.

In population based training, several agents are trained in parallel. When an agent is *ready* (after 5000 episodes), then:

- it may be overwritten by parameters and hyperparameters of another randomly chosen agent, if it is sufficiently better (5000 episode mean capped human normalized score returns are 5% better);
- and independently, the hyperparameters may undergo a change (multiplied by either 1.2 or 1/1.2 with 33% chance).

IMPALA – Architecture

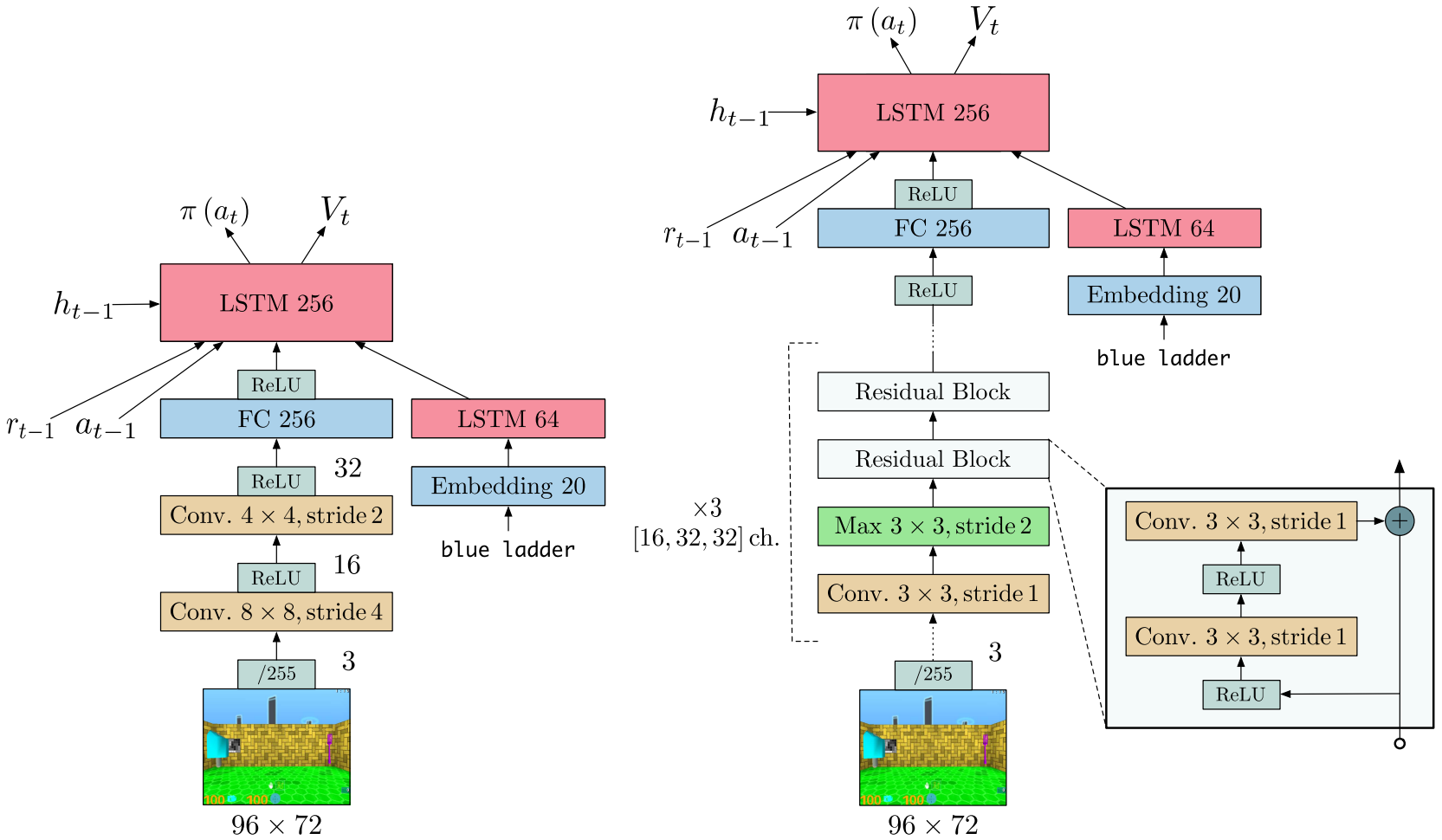


Figure 3 of "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures" by Lasse Espeholt et al.

IMPALA

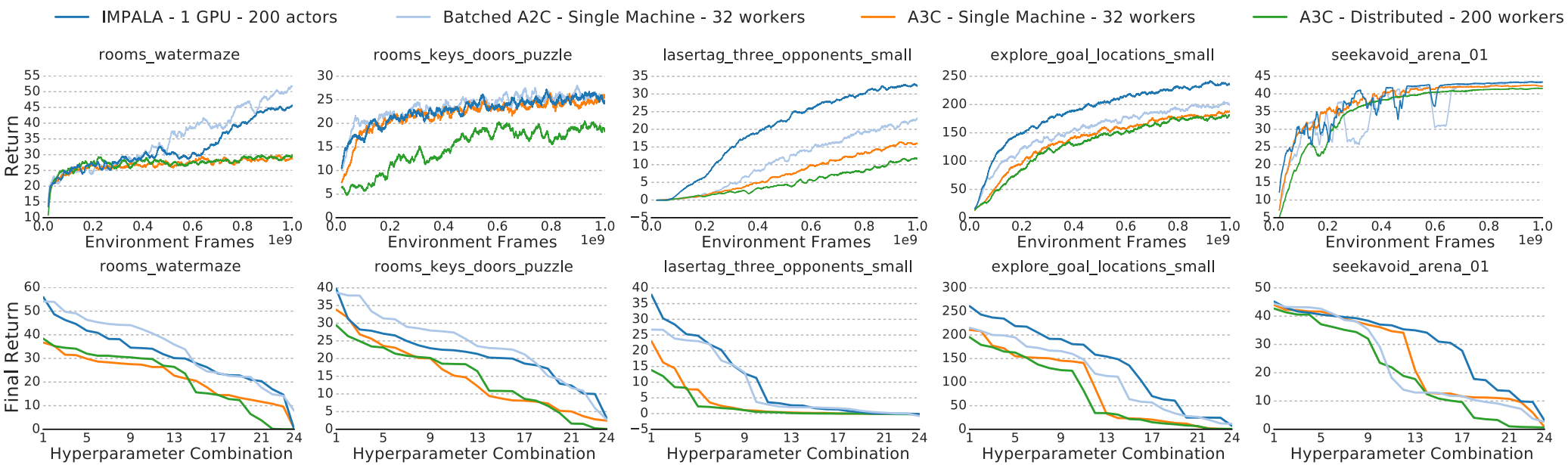
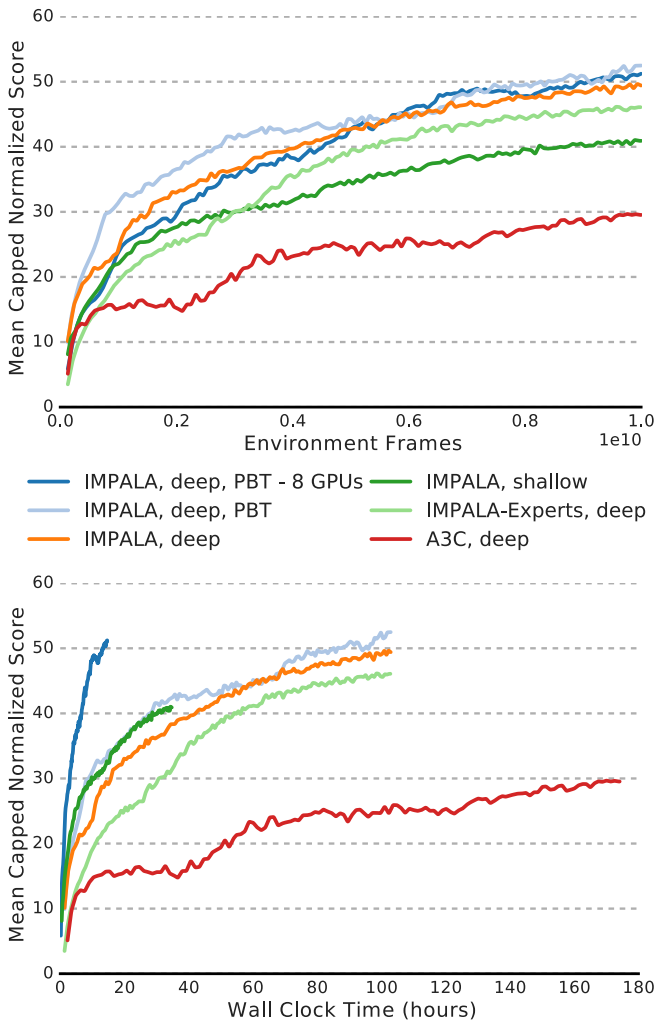


Figure 4 of "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures" by Lasse Espeholt et al.

IMPALA – Learning Curves



Figures 5, 6 of "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures" by Lasse Espeholt et al.

IMPALA – Atari Games

Human Normalised Return	Median	Mean
A3C, shallow, experts	54.9%	285.9%
A3C, deep, experts	117.9%	503.6%
Reactor, experts	187%	N/A
IMPALA, shallow, experts	93.2%	466.4%
IMPALA, deep, experts	191.8%	957.6%
IMPALA, deep, multi-task	59.7%	176.9%

Table 4 of "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures" by Lasse Espeholt et al.

IMPALA – Atari Hyperparameters

Parameter	Value
Image Width	84
Image Height	84
Grayscaling	Yes
Action Repetitions	4
Max-pool over last N action repeat frames	2
Frame Stacking	4
End of episode when life lost	Yes
Reward Clipping	[-1, 1]
Unroll Length (n)	20
Batch size	32
Discount (γ)	0.99
Baseline loss scaling	0.5
Entropy Regularizer	0.01
RMSProp momentum	0.0
RMSProp ε	0.01
Learning rate	0.0006
Clip global gradient norm	40.0
Learning rate schedule	Anneal linearly to 0
	From beginning to end of training.
Population based training (only multi-task agent)	
- Population size	24
- Start parameters	Same as DMLab-30 sweep
- Fitness	Mean capped human normalised scores
	$\left(\sum_l \min\left[1, (s_t - r_t)/(h_t - r_t)\right]\right)/N$
- Adapted parameters	Gradient clipping threshold
	Entropy regularisation
	Learning rate
	RMSProp ε

Table G1 of "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures" by Lasse Espeholt et al.

IMPALA – Ablations

- **No-correction**: no off-policy correction;
- **ε-correction**: add a small value $\varepsilon = 10^{-6}$ during gradient calculation to prevent π to be very small and lead to unstabilities during $\log \pi$ computation;
- **1-step**: no off-policy correction in the update of the value function, TD errors in the policy gradient are multiplied by the corresponding ρ but no cs; it can be considered V-trace "without traces".

	Task 1	Task 2	Task 3	Task 4	Task 5
Without Replay					
V-trace	46.8	32.9	**31.3**	**229.2**	**43.8**
1-Step	**51.8**	**35.9**	25.4	215.8	43.7
ε-correction	44.2	27.3	4.3	107.7	41.5
No-correction	40.3	29.1	5.0	94.9	16.1
With Replay					
V-trace	47.1	**35.8**	**34.5**	**250.8**	**46.9**
1-Step	**54.7**	34.4	26.4	204.8	41.6
ε-correction	30.4	30.2	3.9	101.5	37.6
No-correction	35.0	21.1	2.8	85.0	11.2

Tasks: `rooms_watermaze`, `rooms_keys_doors_puzzle`, `lasertag_three_opponents_small`, `explore_goal_locations_small`, `seekavoid_arena_01`

Table 2 of "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures" by Lasse Espeholt et al.

IMPALA – Ablations

The effect of the policy lag (the number of updates the actor is behind the learned policy) on the performance.

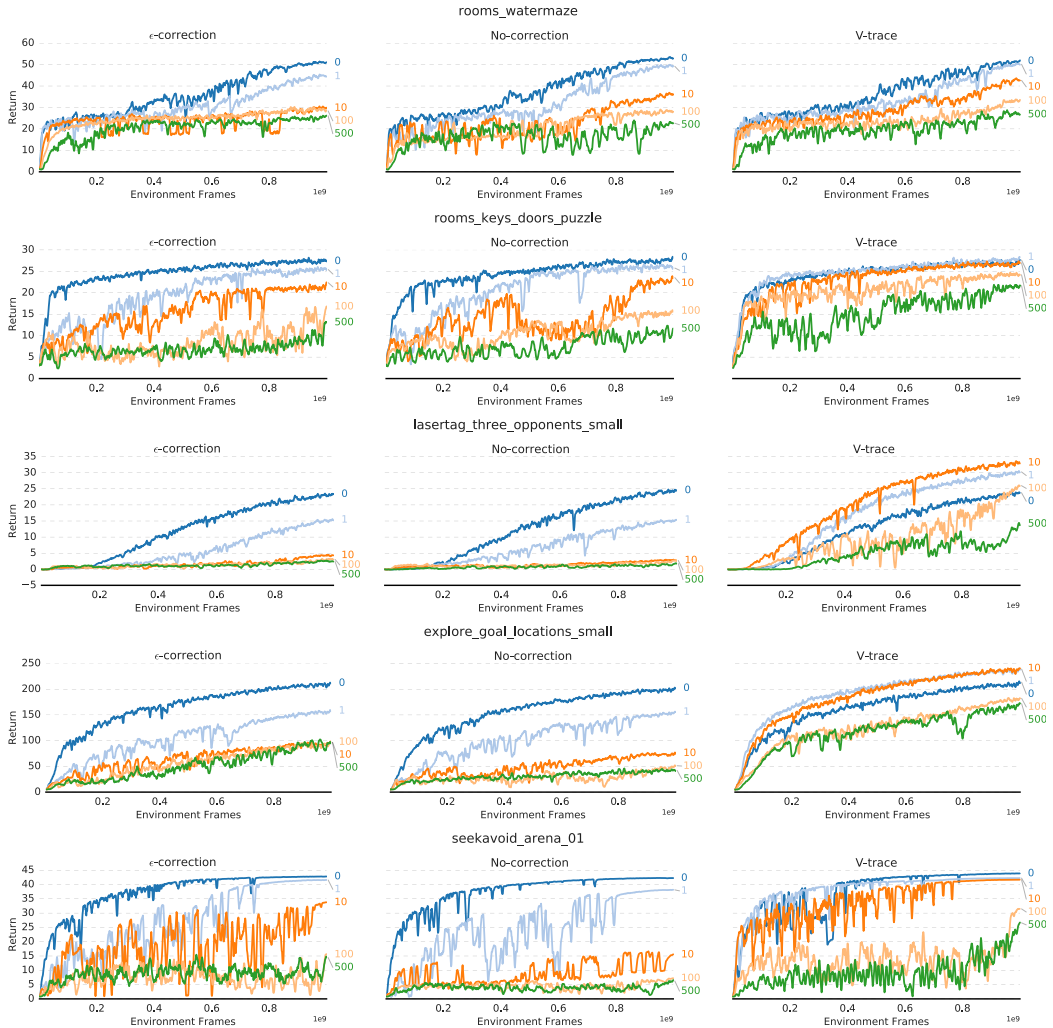


Figure E.1 of "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures" by Lasse Espeholt et al.

PopArt Normalization

An improvement of IMPALA from Sep 2018, which performs normalization of task rewards instead of just reward clipping. PopArt stands for *Preserving Outputs Precisely, while Adaptively Rescaling Targets*.

Assume the value estimate $v(s;\boldsymbol{\theta},\sigma,\mu)$ is computed using a normalized value predictor $n(s;\boldsymbol{\theta})$

$$v(s;\boldsymbol{\theta},\sigma,\mu) \stackrel{\text{def}}{=} \sigma n(s;\boldsymbol{\theta}) + \mu,$$

and further assume that $n(s;\boldsymbol{\theta})$ is an output of a linear function

$$n(s;\boldsymbol{\theta}) \stackrel{\text{def}}{=} \boldsymbol{\omega}^T f(s;\boldsymbol{\theta} - \{\boldsymbol{\omega}, b\}) + b.$$

We can update the σ and μ using exponentially moving average with decay rate β (in the paper, first moment μ and second moment υ is tracked, and the standard deviation is computed as $\sigma = \sqrt{\upsilon - \mu^2}$; decay rate $\beta = 3 \cdot 10^{-4}$ is employed).

PopArt Normalization

Utilizing the parameters μ and σ, we can normalize the observed (unnormalized) returns as $(G - \mu)/\sigma$, and use an actor-critic algorithm with advantage $(G - \mu)/\sigma - n(S; \boldsymbol{\theta})$.

However, in order to make sure the value function estimate does not change when the normalization parameters change, the parameters $\boldsymbol{\omega}, b$ used to compute the value estimate

$$v(s; \boldsymbol{\theta}, \sigma, \mu) \stackrel{\text{def}}{=} \sigma \cdot \Big(\boldsymbol{\omega}^T f(s; \boldsymbol{\theta} - \{\boldsymbol{\omega}, b\}) + b\Big) + \mu$$

are updated under any change $\mu \to \mu'$ and $\sigma \to \sigma'$ as

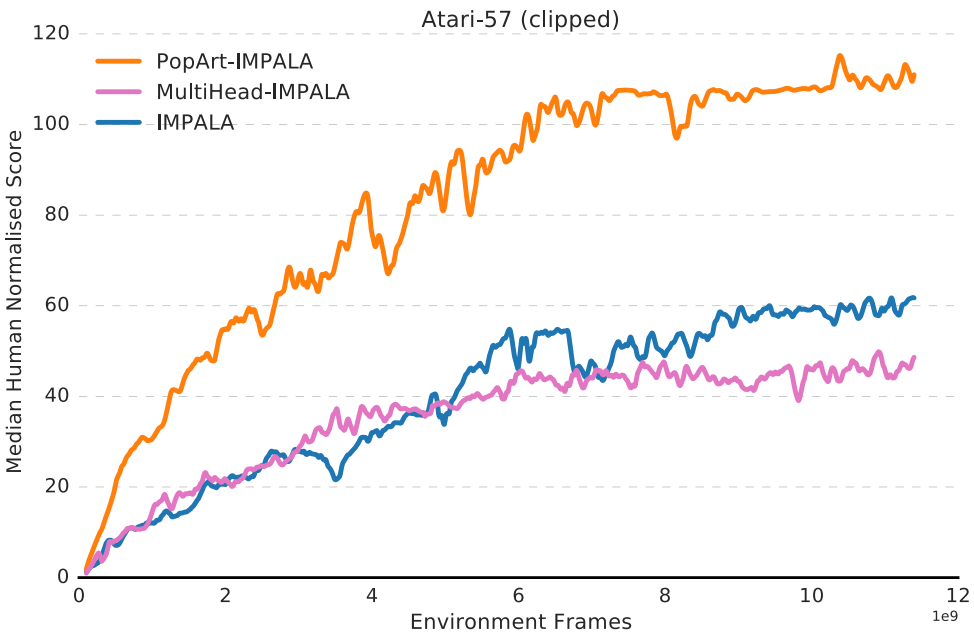
$$\boldsymbol{\omega}' \leftarrow \frac{\sigma}{\sigma'}\boldsymbol{\omega},$$
$$b' \leftarrow \frac{\sigma b + \mu - \mu'}{\sigma'}.$$

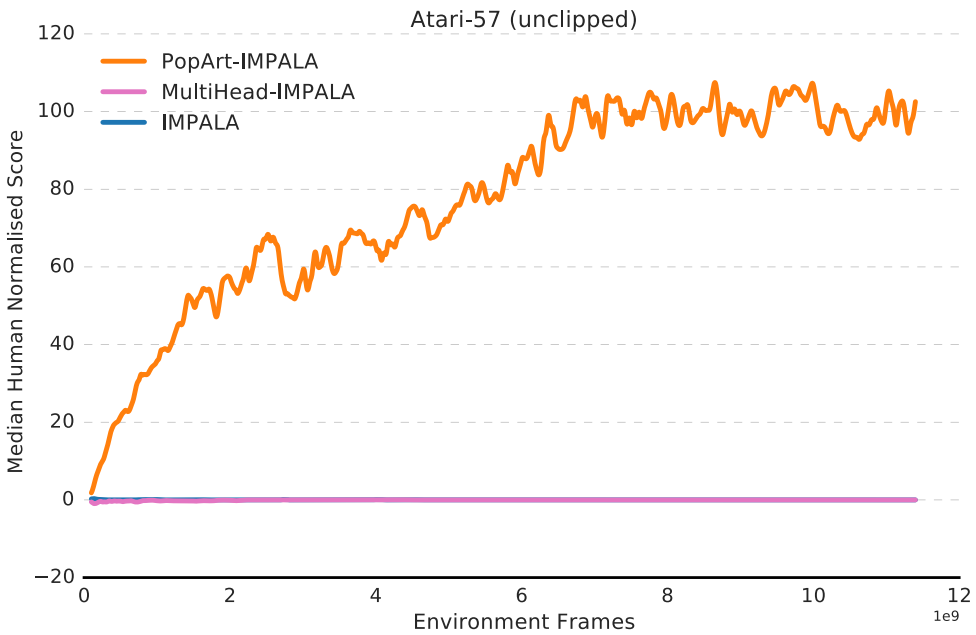
In multi-task settings, we train a task-agnostic policy and task-specific value functions (therefore, $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, and $\boldsymbol{n}(s; \boldsymbol{\theta})$ are vectors).

PopArt Results

	Atari-57		Atari-57 (unclipped)		DmLab-30	
Agent	Random	Human	Random	Human	Train	Test
IMPALA	59.7%	28.5%	0.3%	1.0%	60.6%	58.4%
PopArt-IMPALA	110.7%	101.5%	107.0%	93.7%	73.5%	72.8%

Table 1 of "Multi-task Deep Reinforcement Learning with PopArt" by Matteo Hessel et al.





Figures 1, 2 of "Multi-task Deep Reinforcement Learning with PopArt" by Matteo Hessel et al.

PopArt Results

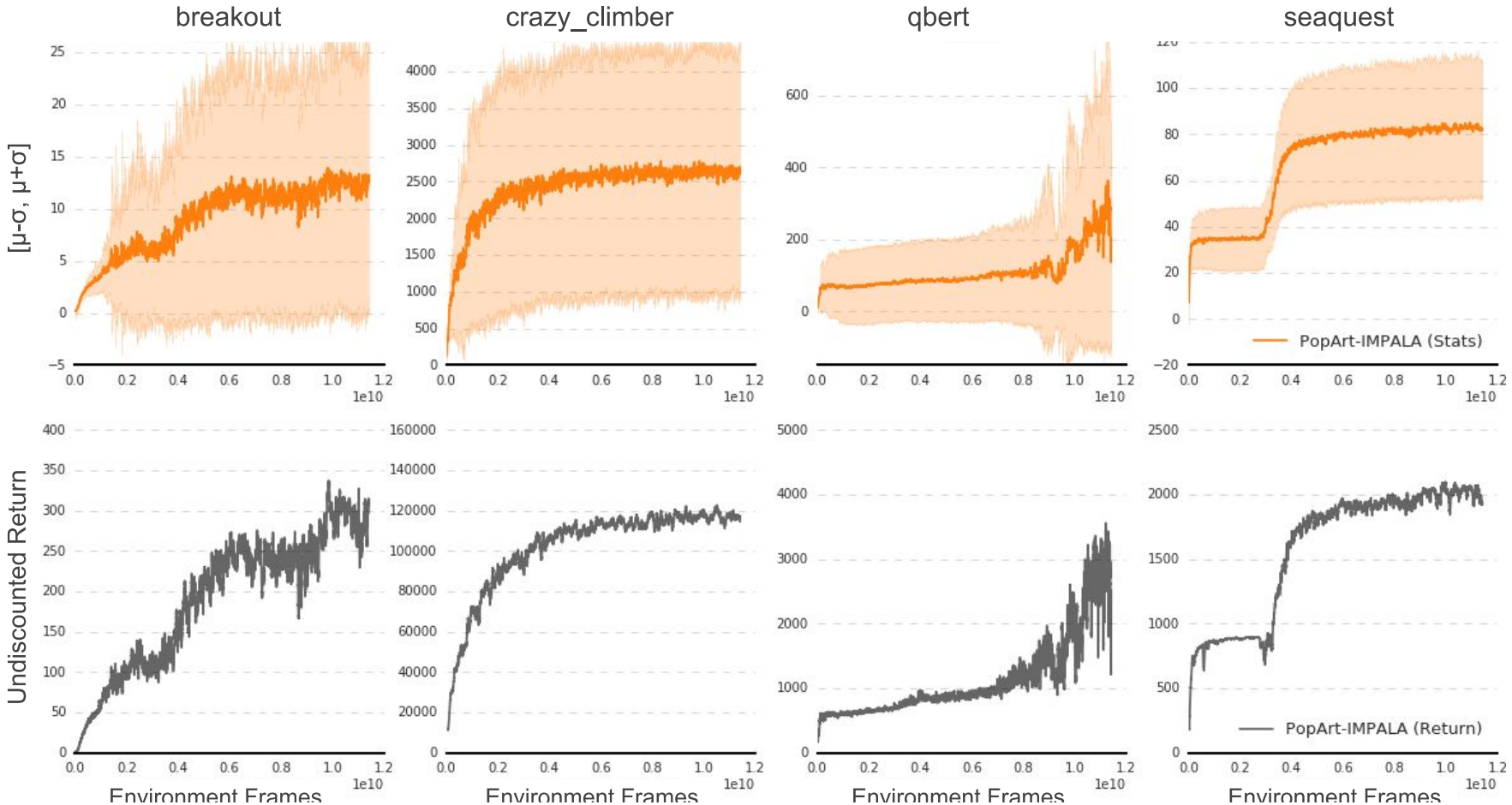


Figure 3 of "Multi-task Deep Reinforcement Learning with PopArt" by Matteo Hessel et al.

Normalization statistics on chosen environments.

Transformed Rewards

So far, we have clipped the rewards in DQN on Atari environments.

Consider a Bellman operator $\mathcal{T}$

$$(\mathcal{T}q)(s,a) \stackrel{\text{def}}{=} \mathbb{E}_{s',r\sim p}\Big[r + \gamma \max_{a'} q(s',a')\Big].$$

Instead of clipping the magnitude of rewards, we might use a function $h : \mathbb{R} \to \mathbb{R}$ to reduce their scale. We define a transformed Bellman operator $\mathcal{T}_h$ as

$$(\mathcal{T}_h q)(s,a) \stackrel{\text{def}}{=} \mathbb{E}_{s',r\sim p}\Big[h\Big(r + \gamma \max_{a'} h^{-1}\big(q(s',a')\big)\Big)\Big].$$

Transformed Rewards

It is easy to prove the following two propositions from a 2018 paper *Observe and Look Further: Achieving Consistent Performance on Atari* by Tobias Pohlen et al.

1. If $h(z) = \alpha z$ for $\alpha > 0$, then $\mathcal{T}_h^k q \xrightarrow{k\to\infty} h \circ q_* = \alpha q_*$.

   The statement follows from the fact that it is equivalent to scaling the rewards by a constant α.

2. When h is strictly monotonically increasing and the MDP is deterministic, then $\mathcal{T}_h^k q \xrightarrow{k\to\infty} h \circ q_*$.

   This second proposition follows from

$$h \circ q_* = h \circ \mathcal{T} q_* = h \circ \mathcal{T}(h^{-1} \circ h \circ q_*) = \mathcal{T}_h(h \circ q_*),$$

   where the last equality only holds if the MDP is deterministic.

Transformed Rewards

For stochastic MDP, the authors prove that if h is strictly monotonically increasing, Lipschitz continuous with Lipschitz constant L_h, and has a Lipschitz continuous inverse with Lipschitz constant $L_{h^{-1}}$, then for $\gamma < \frac{1}{L_h L_{h^{-1}}}$, $\mathcal{T}_h$ is again a contraction. (Proof in Proposition A.1.)

For the Atari environments, the authors propose the transformation

$$h(x) \stackrel{\text{def}}{=} \operatorname{sign}(x)\left(\sqrt{|x|+1}-1\right)+\varepsilon x$$

with $\varepsilon = 10^{-2}$. The additive regularization term ensures that h^{-1} is Lipschitz continuous.

It is straightforward to verify that

$$h^{-1}(x) = \operatorname{sign}(x)\left(\left(\frac{\sqrt{1+4\varepsilon(|x|+1+\varepsilon)}-1}{2\varepsilon}\right)^2-1\right).$$

In practice, discount factor larger than $\frac{1}{L_h L_{h^{-1}}}$ is being used – however, it seems to work.

Recurrent Replay Distributed DQN (R2D2)

Proposed in 2019, to study the effects of recurrent state, experience replay and distributed training.

R2D2 utilizes prioritized replay, n-step double Q-learning with $n = 5$, convolutional layers followed by a 512-dimensional LSTM passed to duelling architecture, generating experience by a large number of actors (256; each performing approximately 260 steps per second) and learning from batches by a single learner (achieving 5 updates per second using mini-batches of 64 sequences of length 80).

Rewards are transformed instead of clipped, and no loss-of-life-as-episode-end heuristic is used.

Instead of individual transitions, the replay buffer consists of fixed-length ($m = 80$) sequences of (s, a, r), with adjacent sequences overlapping by 40 time steps.

The prioritized replay employs a combination of the maximum and the average absolute 5-step TD errors δ_i over the sequence: $p = \eta \max_i \delta_i + (1 - \eta)\overline{\delta}$, for η and the priority exponent set to 0.9.

Recurrent Replay Distributed DQN (R2D2)

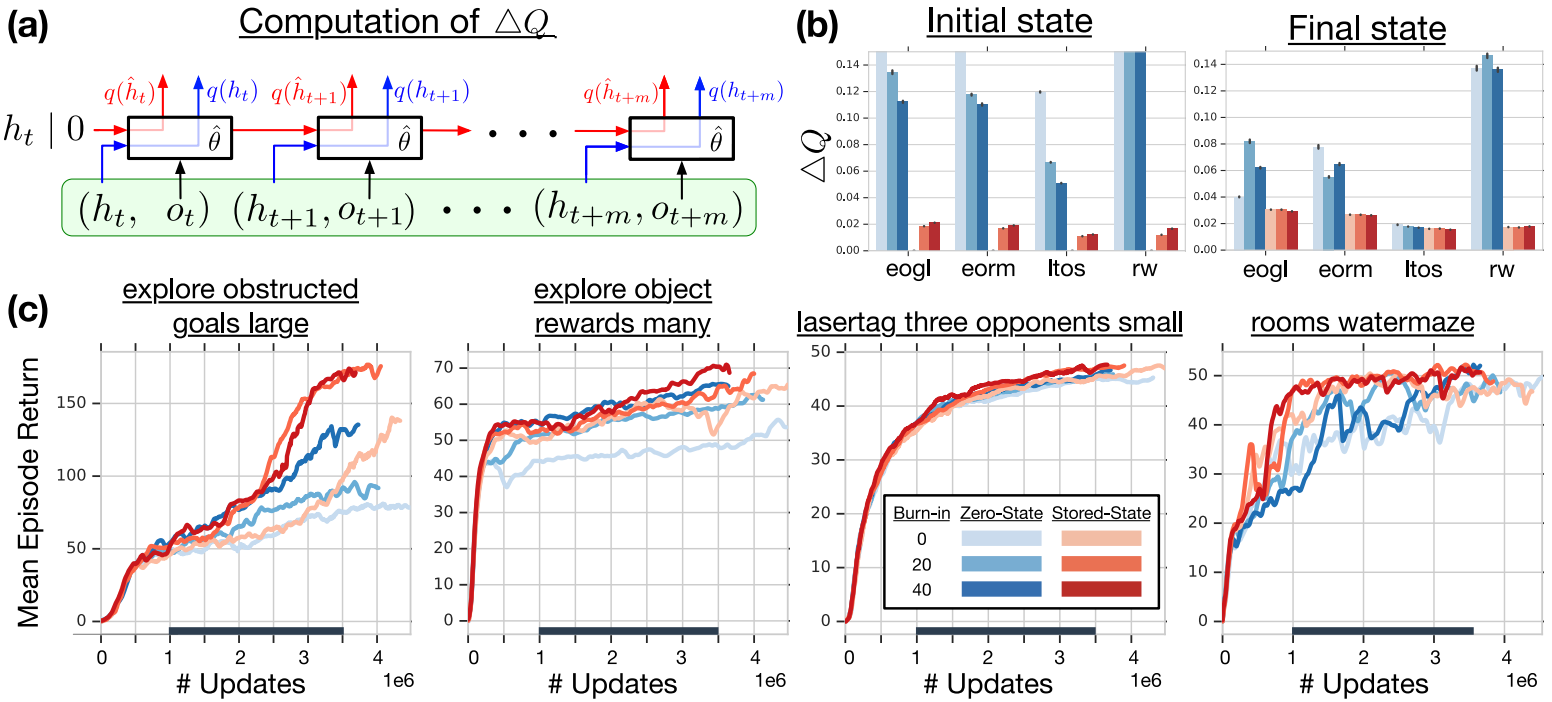


Figure 1: Top row shows Q-value discrepancy ΔQ as a measure for recurrent state staleness. **(a)** Diagram of how ΔQ is computed, with green box indicating a whole sequence sampled from replay. For simplicity, $l = 0$ (no burn-in). **(b)** ΔQ measured at first state and last state of replay sequences, for agents training on a selection of DMLab levels (indicated by initials) with different training strategies. Bars are averages over seeds and through time indicated by bold line on x-axis in bottom row. **(c)** Learning curves on the same levels, varying the training strategy, and averaged over 3 seeds.

Figure 1 of "Recurrent Experience Replay in Distributed Reinforcement Learning" by Steven Kapturowski et al.

Recurrent Replay Distributed DQN (R2D2)

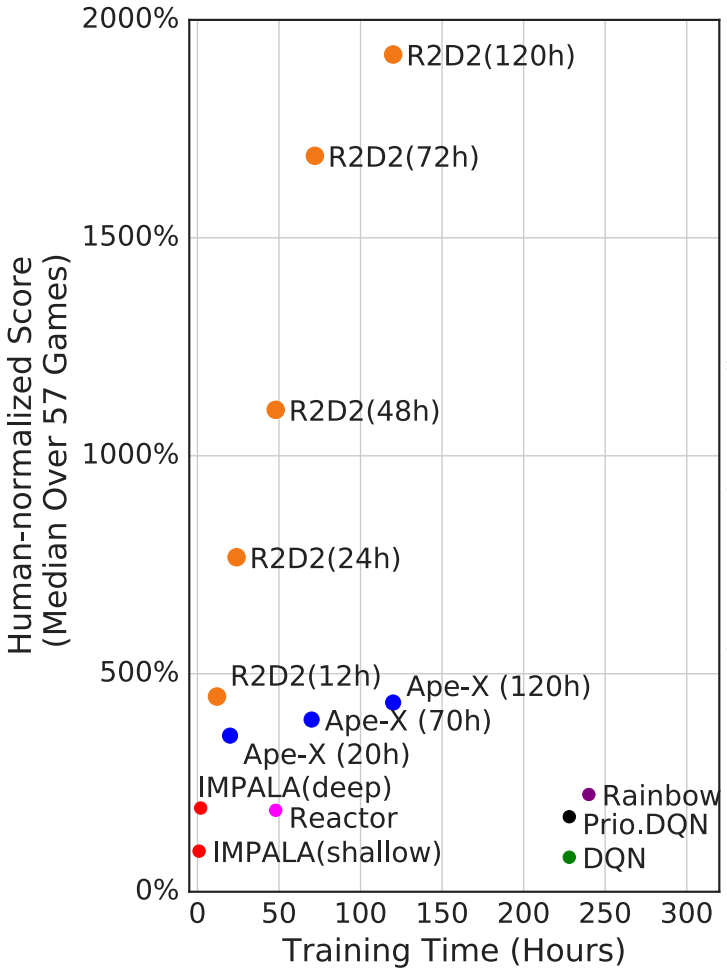


Figure 2 of "Recurrent Experience Replay in Distributed Reinforcement Learning" by Steven Kapturowski et al.

	Atari-57		DMLab-30	
Human-Normalized Score	**Median**	**Mean**	**Median**	**Mean-Capped**
Ape-X (Horgan et al., 2018)	434.1%	1695.6%	–	–
Reactor (Gruslys et al., 2018)	187.0%	–	–	–
IMPALA, deep (Espeholt et al., 2018)	191.8%	957.6%	49.0%	45.8%
IMPALA, shallow (re-run)	–	–	89.7%	73.6%
IMPALA, deep (re-run)	–	–	**107.5%**	85.1%
R2D2+	–	–	99.5%	**85.7%**
R2D2, feed-forward	589.2%	1974.4%	–	–
R2D2	**1920.6%**	**4024.9%**	96.9%	78.3%

Table 1 of "Recurrent Experience Replay in Distributed Reinforcement Learning" by Steven Kapturowski et al.

Recurrent Replay Distributed DQN (R2D2)

Number of actors	256
Actor parameter update interval	400 environment steps
Sequence length m	80 (+ prefix of $l = 40$ in burn-in experiments)
Replay buffer size	4×10^6 observations (10^5 part-overlapping sequences)
Priority exponent	0.9
Importance sampling exponent	0.6
Discount γ	0.997
Minibatch size	64 (32 for R2D2+ on DMLab)
Optimizer	Adam (Kingma & Ba, 2014)
Optimizer settings	learning rate $= 10^{-4}$, $\varepsilon = 10^{-3}$
Target network update interval	2500 updates
Value function rescaling	$h(x) = \text{sign}(x)(\sqrt{\lvert x\rvert + 1} - 1) + \epsilon x,\ \epsilon = 10^{-3}$

Table 2: Hyper-parameters values used in R2D2. All missing parameters follow the ones in Ape-X (Horgan et al., 2018).

Table 2 of "Recurrent Experience Replay in Distributed Reinforcement Learning" by Steven Kapturowski et al.

Recurrent Replay Distributed DQN (R2D2)

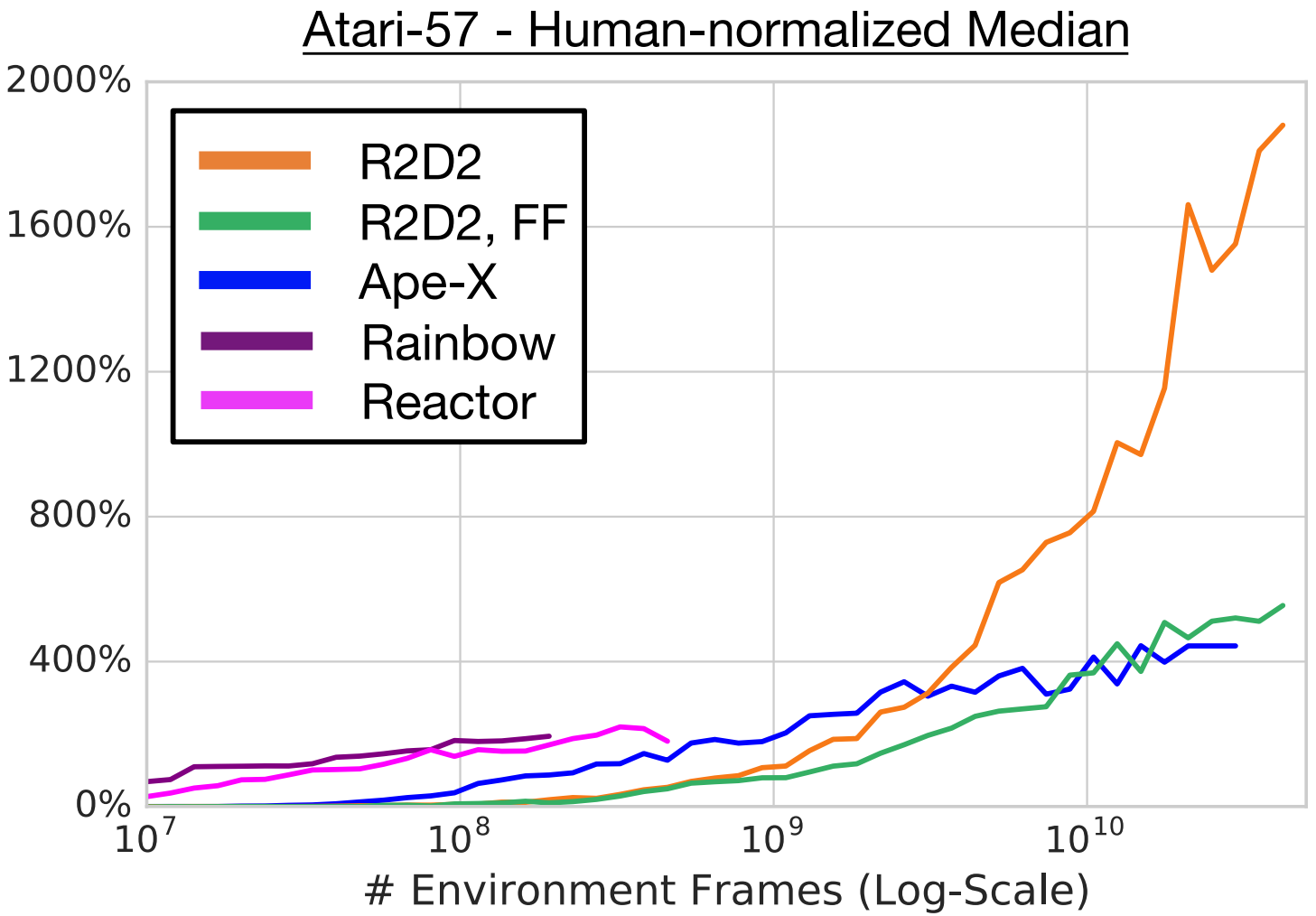


Figure 9 of "Recurrent Experience Replay in Distributed Reinforcement Learning" by Steven Kapturowski et al.

Recurrent Replay Distributed DQN (R2D2)

Ablations comparing the reward clipping instead of value rescaling (**Clipped**), smaller discount factor $\gamma = 0.99$ (**Discount**) and **Feed-Forward** variant of R2D2. Furthermore, life-loss **reset** evaluates resetting an episode on life loss, with **roll** preventing value bootstrapping (but not LSTM unrolling).

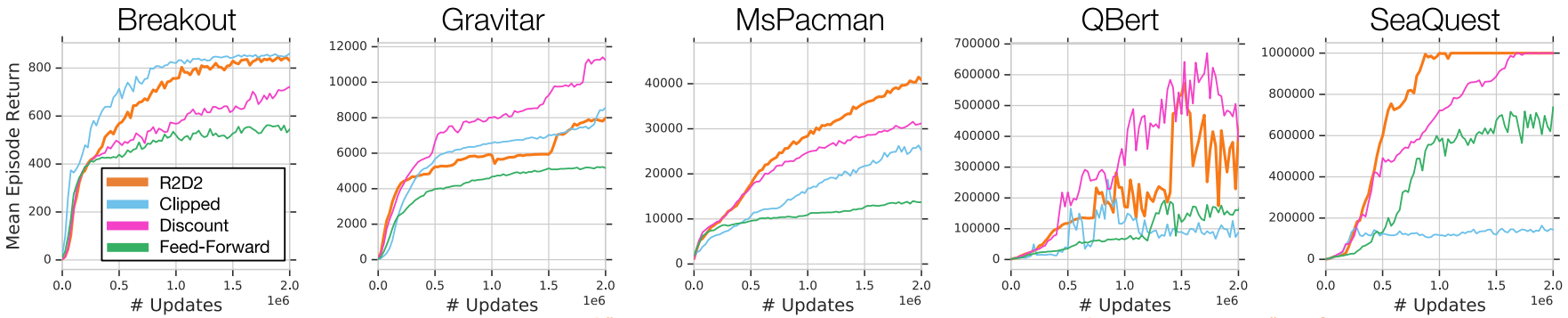


Figure 4 of "Recurrent Experience Replay in Distributed Reinforcement Learning" by Steven Kapturowski et al.

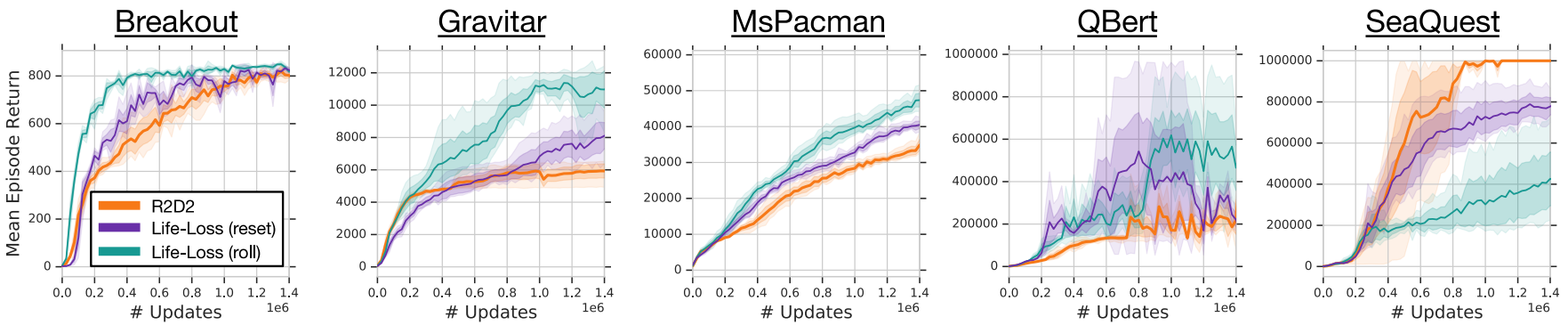


Figure 7 of "Recurrent Experience Replay in Distributed Reinforcement Learning" by Steven Kapturowski et al.

Utilization of LSTM Memory During Inference

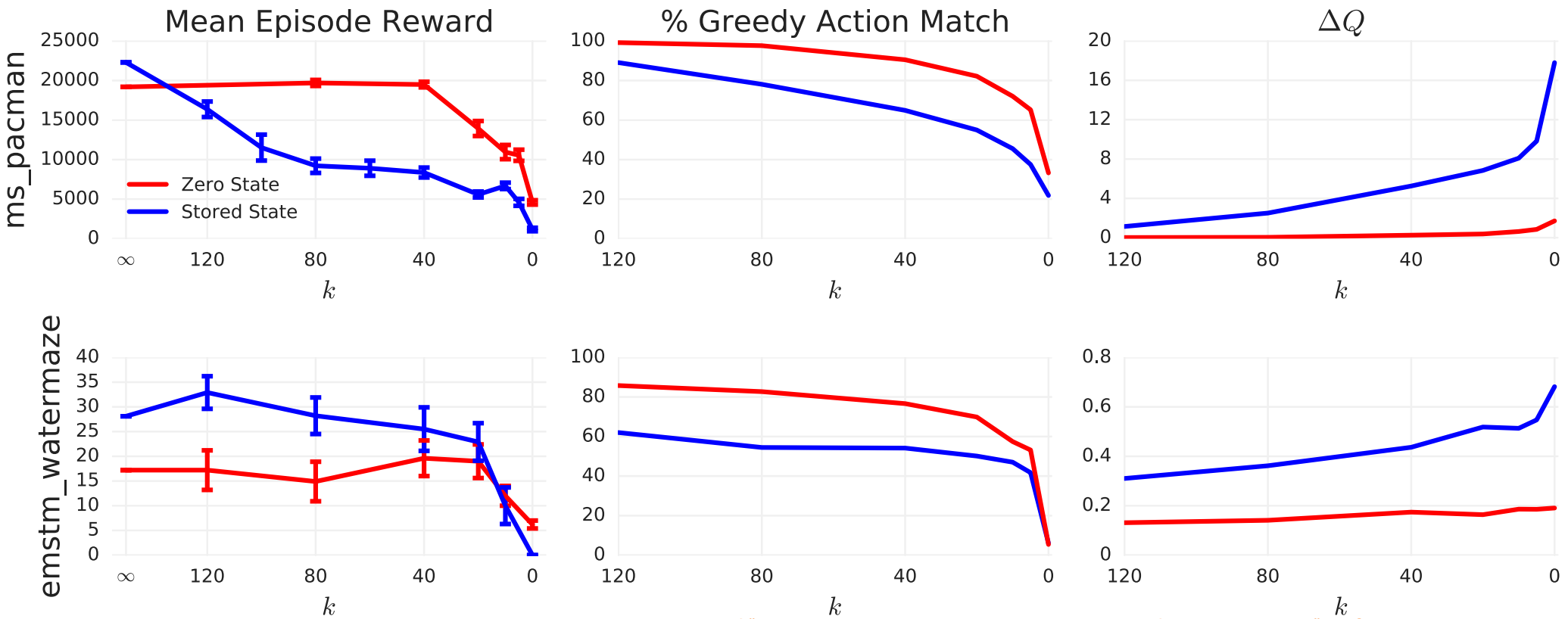


Figure 5 of "Recurrent Experience Replay in Distributed Reinforcement Learning" by Steven Kapturowski et al.

Agent57

The Agent57 is an agent (from Mar 2020) capable of outperforming the standard human benchmark on all 57 games.

Its most important components are:

- Retrace; from *Safe and Efficient Off-Policy Reinforcement Learning* by Munos et al., https://arxiv.org/abs/1606.02647,
- Never give up strategy; from *Never Give Up: Learning Directed Exploration Strategies* by Badia et al., https://arxiv.org/abs/2002.06038,
- Agent57 itself; from *Agent57: Outperforming the Atari Human Benchmark* by Badia et al., https://arxiv.org/abs/2003.13350.

Retrace

$$\mathcal{R}q(s,a) \stackrel{\text{def}}{=} q(s,a) + \mathbb{E}_b\left[\sum_{t\geq 0}\gamma^t\left(\prod_{j=1}^{t} c_t\right)\left(R_{t+1} + \gamma\mathbb{E}_{A_{t+1}\sim\pi}q(S_{t+1},A_{t+1}) - q(S_t,A_t)\right)\right],$$

where there are several possibilities for defining the traces c_t:

- **importance sampling**, $c_t = \rho_t = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$,
  - the usual off-policy correction, but with possibly very high variance,
  - note that $c_t = 1$ in the on-policy case;
- **Tree-backup TB(λ)**, $c_t = \lambda\pi(A_t|S_t)$,
  - the Tree-backup algorithm extended with traces,
  - however, c_t can be much smaller than 1 in the on-policy case;
- **Retrace(λ)**, $c_t = \lambda \min\left(1, \frac{\pi(A_t|S_t)}{b(A_t|S_t)}\right)$,
  - off-policy correction with limited variance, with $c_t = 1$ in the on-policy case.

The authors prove that $\mathcal{R}$ has a unique fixed point q_π for any $0 \leq c_t \leq \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$.

Never Give Up

The NGU (Never Give Up) agent performs *curiosity-driver exploration*, and augment the extrinsic (MDP) rewards with an intrinsic reward. The augmented reward at time t is then $r_t^\beta \stackrel{\text{def}}{=} r_t^e + \beta r_t^i$, with β a scalar weight of the intrinsic reward.

The intrinsic reward fulfills three goals:

1. quickly discourage visits of the same state in the same episode;
2. slowly discourage visits of the states visited many times in all episodes;
3. ignore the parts of the state not influenced by the agent's actions.

The intrinsic rewards is a combination of the episodic novelty r_t^{episodic} and life-long novelty α_t:

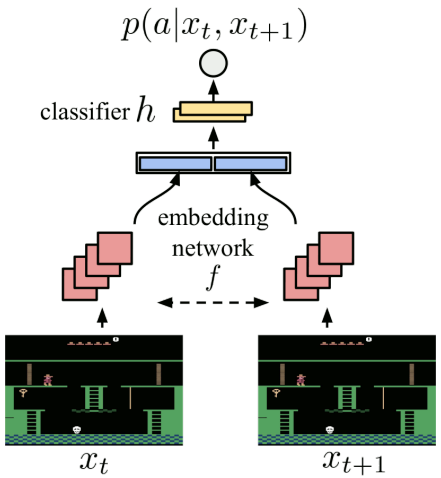
$$r_t^i \stackrel{\text{def}}{=} r_t^{\text{episodic}} \cdot \text{clip}\left(1 \leq \alpha_t \leq L = 5\right).$$

Never Give Up

The episodic novelty works by storing the embedded states $f(S_t)$ visited during the episode in episodic memory M.

The r_t^{episodic} is then estimated as

$$r_t^{\text{episodic}} = \frac{1}{\sqrt{\text{visit count of } f(S_t)}}.$$



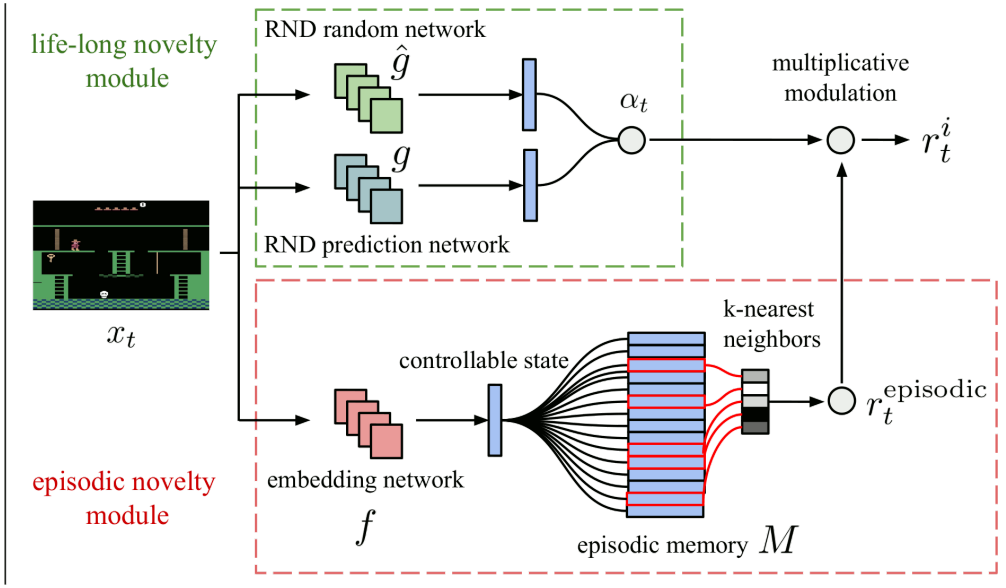


Figure 1 of "Never Give Up: Learning Directed Exploration Strategies" by A. P. Badia et al.

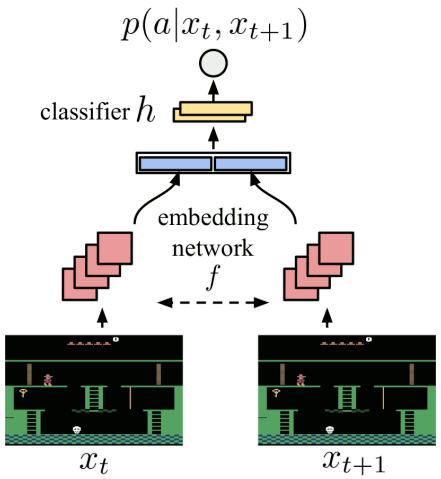
The visit count is estimated using similarities of k-nearest neighbors of $f(S_t)$ measured via an inverse kernel $K(x, z) = \frac{\varepsilon}{\frac{d(x,z)^2}{d_m^2} + \varepsilon}$ for d_m a running mean of the k-nearest neighbor distance:

$$r_t^{\text{episodic}} = \frac{1}{\sqrt{\sum_{f_i \in N_k} K(f(S_t), f_i)} + c}, \text{ with pseudo-count c=0.001.}$$

Never Give Up

The state embeddings are trained to ignore the parts not influenced by the actions of the agent.

To this end, Siamese network f is trained to predict $p(A_t|S_t, S_{t+1})$, i.e., the action A_t taken by the agent in state S_t when the resulting state is S_{t+1}.



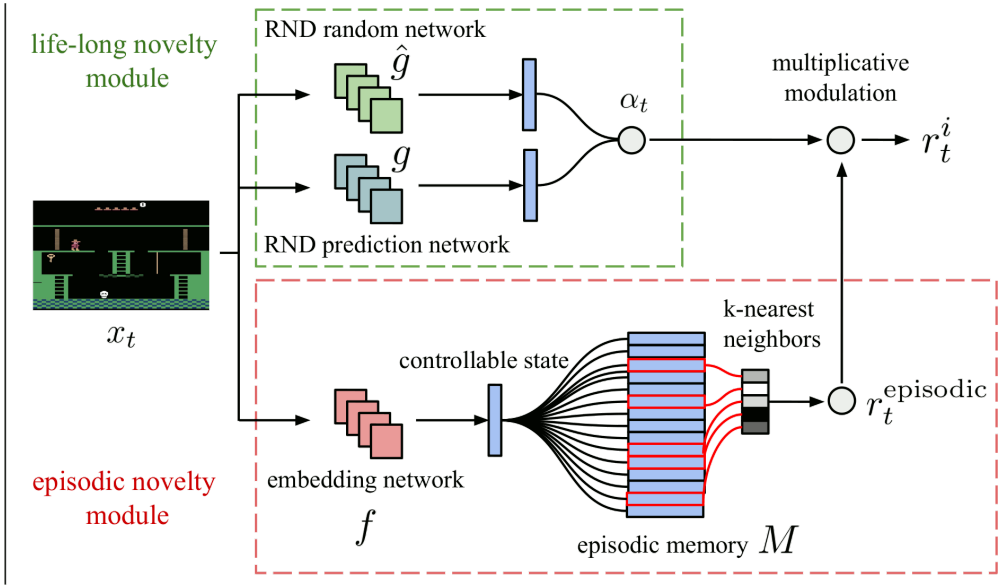


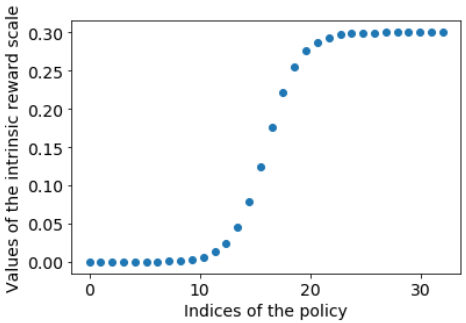
Figure 1 of "Never Give Up: Learning Directed Exploration Strategies" by A. P. Badia et al.

The life-long novelty $\alpha_t = 1 + \frac{\|\hat{g}-g\|^2 - \mu_{\text{err}}}{\sigma_{\text{err}}}$ is trained using random network distillation (RND), where a predictor network $\hat{g}$ tries to predict the output of an untrained convolutional network g by minimizing the mean squared error; the μ_{err} and σ_{err} are the running mean and standard deviation of the error $\|\hat{g} - g\|^2$.

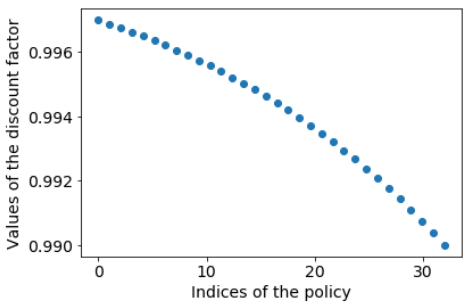
Never Give Up

The NGU agent uses transformed Retrace loss with the augmented reward

$$r_t^i \overset{\text{def}}{=} r_t^{\text{episodic}} \cdot \text{clip}\left(1 \leq \alpha_t \leq L = 5\right).$$



(a) Values taken by the $\{\beta_i\}_{i=0}^{N-1}$

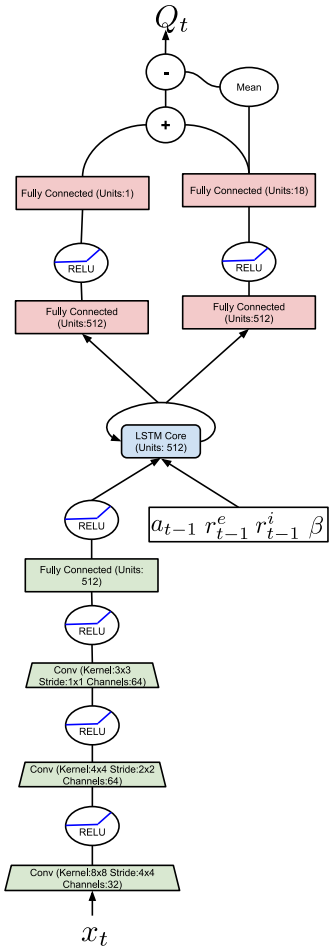


(b) Values taken by the $\{\gamma_i\}_{i=0}^{N-1}$

Figure 7b of "Never Give Up: Learning Directed Exploration Strategies" by A. P. Badia et al.

To support multiple policies concentrating either on the extrinsic or the intrinsic reward, the NGU agent trains a parametrized action-value function $q(s, a, \beta_i)$ which corresponds to reward $r_t^{\beta_i}$ for $\beta_0 = 0$ and $\gamma_0 = 0.997$, ..., $\beta_{N-1} = \beta$ and $\gamma_{N-1} = 0.99$.

For evaluation, $q(s, a, 0)$ is employed.



(b) Detailed R2D2 Agent

Figure 17 of "Never Give Up: Learning Directed Exploration Strategies" by A. P. Badia et al.

Never Give Up

Algorithm	Gravitar	MR	Pitfall!	PrivateEye	Solaris	Venture
Human	3.4k	4.8k	6.5k	69.6k	**12.3k**	1.2k
Best baseline	**15.7k**	**11.6k**	0.0	11k	5.5k	2.0k
RND	3.9k	10.1k	-3	8.7k	3.3k	1.9k
R2D2+RND	15.6k±0.6k	10.4k±1.2k	-0.5±0.3	19.5k±3.5k	4.3k±0.6k	**2.7k±0.0k**
R2D2(Retrace)	13.3k±0.6k	2.3k±0.4k	-3.5±1.2	32.5k±4.7k	6.0k±1.1k	2.0k±0.0k
NGU(N=1)-RND	12.4k±0.8k	3.0k±0.0k	**15.2k±9.4k**	40.6k±0.0k	5.7k±1.8k	46.4±37.9
NGU(N=1)	11.0k±0.7k	8.7k±1.2k	9.4k±2.2k	60.6k±16.3k	5.9k±1.6k	876.3±114.5
NGU(N=32)	14.1k±0.5k	10.4k±1.6k	8.4k±4.5k	**100.0k±0.4k**	4.9k±0.3k	1.7k±0.1k

Table 1: Results against exploration algorithm baselines. Best baseline takes the best result among R2D2 (Kapturowski et al., 2019), DQN + PixelCNN (Ostrovski et al., 2017), DQN + CTS (Bellemare et al., 2016), RND (Burda et al., 2018b), and PPO + CoEx (Choi et al., 2018) for each game.

Table 1 of "Never Give Up: Learning Directed Exploration Strategies" by A. P. Badia et al.

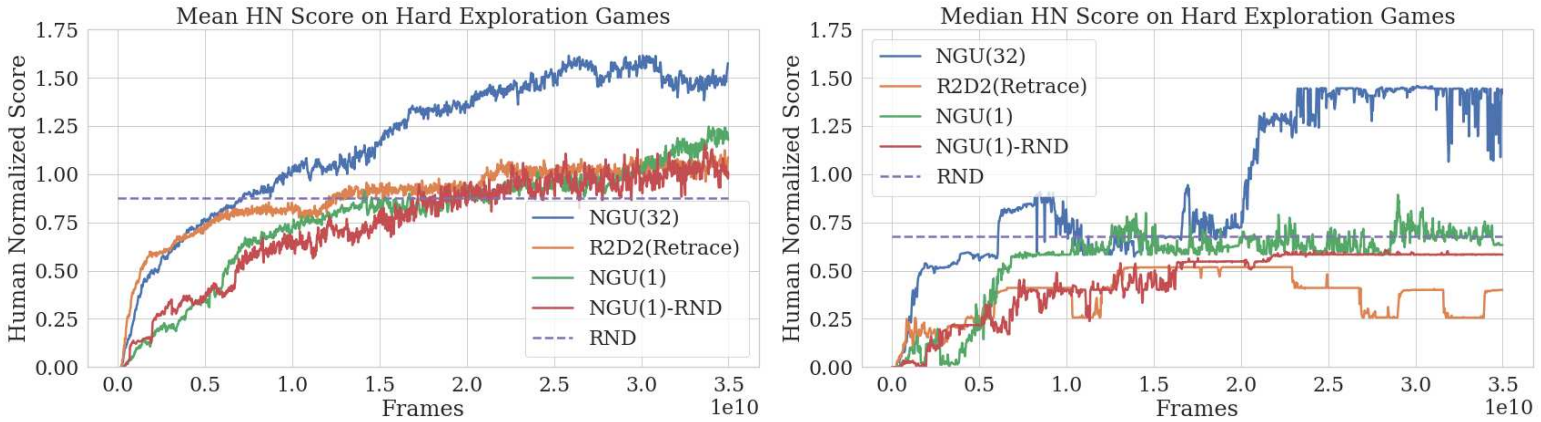


Figure 4: Human Normalized Scores on the 6 hard exploration games.

Figure 4 of "Never Give Up: Learning Directed Exploration Strategies" by A. P. Badia et al.

Never Give Up Ablations

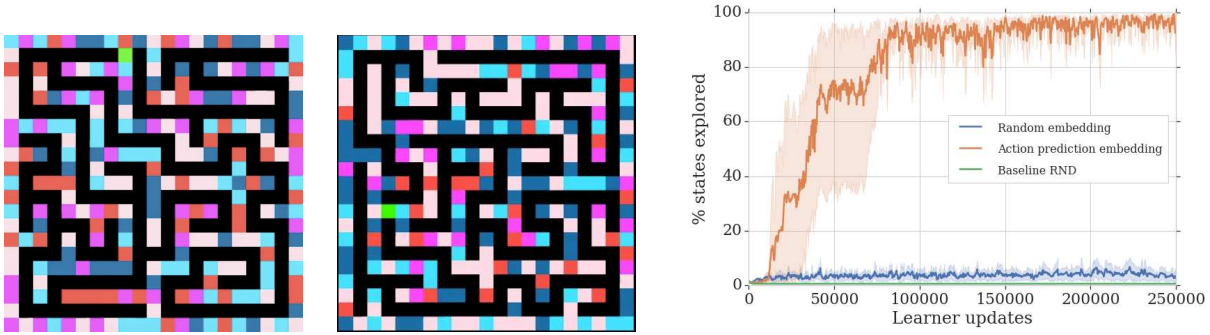


Figure 2: (Left and Center) Sample screens of Random Disco Maze. The agent is in green, and pathways in black. The colors of the wall change at every time step. (Right) Learning curves for Random projections vs. learned controllable states and a baseline RND implementation.

Figure 2 of "Never Give Up: Learning Directed Exploration Strategies" by A. P. Badia et al.

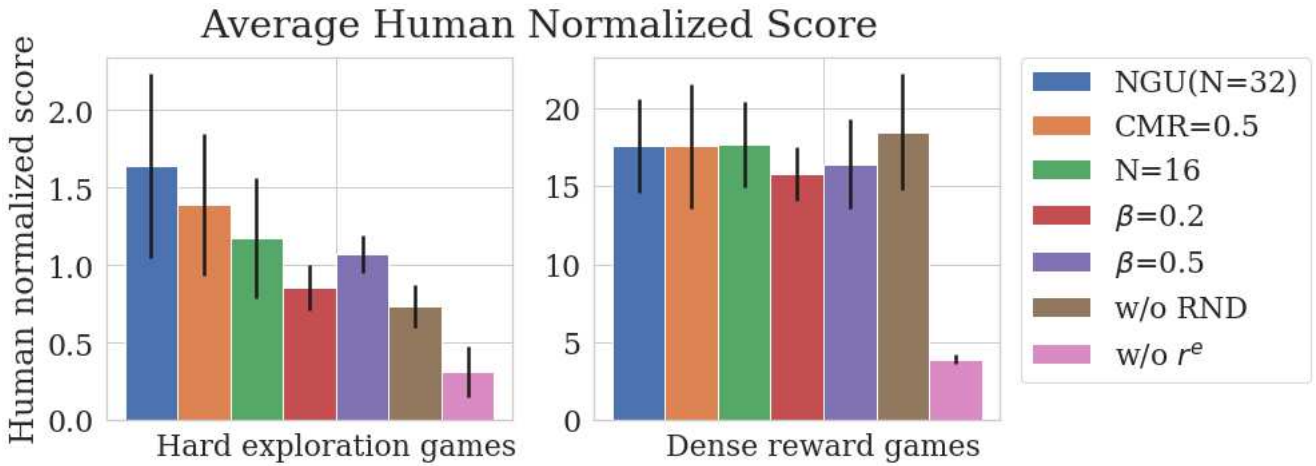


Figure 3 of "Never Give Up: Learning Directed Exploration Strategies" by A. P. Badia et al.

Then Agent57 improves NGU with:

- splitting the action-value as $q(s, a, j; \boldsymbol{\theta}) \stackrel{\text{def}}{=} q(s, a, j; \boldsymbol{\theta}^e) + \beta_j q(s, a, j; \boldsymbol{\theta}^i)$, where
  - $q(s, a, j; \boldsymbol{\theta}^e)$ is trained with r_e as targets, and
  - $q(s, a, j; \boldsymbol{\theta}^i)$ is trained with r_i as targets.
- instead of considering all (β_j, γ_j) equal, we train a meta-controller using a non-stationary multi-arm bandit algorithm, where arms correspond to the choice of j for a whole episode (so an actor first samples a j using multi-arm bandit problem and then updates it according to the observed return), and the reward signal is the undiscounted extrinsic episode return; each actor uses a different level of ε_l-greedy behavior;
- γ_{N-1} is increased from 0.997 to 0.9999.

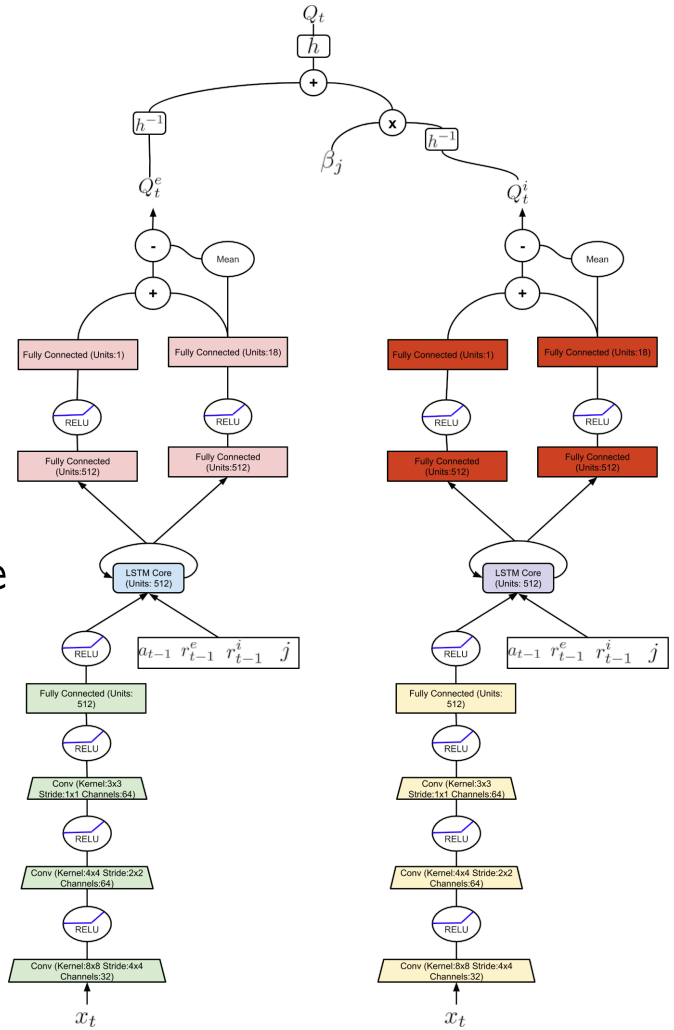


Figure 10 of "Agent57: Outperforming the Atari Human Benchmark" by A. P. Badia et al.

Agent57 – Results

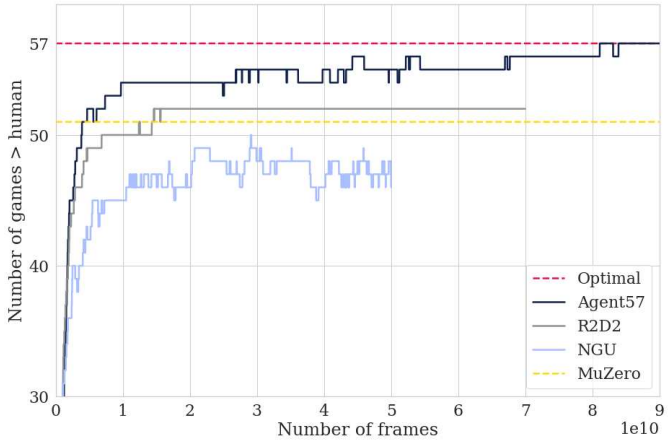


Figure 1. Number of games where algorithms are better than the human benchmark throughout training for Agent57 and state-of-the-art baselines on the 57 Atari games.

Figure 1 of "Agent57: Outperforming the Atari Human Benchmark" by A. P. Badia et al.

Table 1. Number of games above human, mean capped, mean and median human normalized scores for the 57 Atari games.

Statistics	Agent57	R2D2 (bandit)	NGU	R2D2 (Retrace)	R2D2	MuZero
Capped mean	**100.00**	96.93	95.07	94.20	94.33	89.92
Number of games > human	**57**	54	51	52	52	51
Mean	4766.25	5461.66	3421.80	3518.36	4622.09	**5661.84**
Median	1933.49	2357.92	1359.78	1457.63	1935.86	**2381.51**
40th Percentile	1091.07	**1298.80**	610.44	817.77	1176.05	1172.90
30th Percentile	614.65	**648.17**	267.10	420.67	529.23	503.05
20th Percentile	**324.78**	303.61	226.43	267.25	215.31	171.39
10th Percentile	**184.35**	116.82	107.78	116.03	115.33	75.74
5th Percentile	**116.67**	93.25	64.10	48.32	50.27	0.03

Table 1 of "Agent57: Outperforming the Atari Human Benchmark" by A. P. Badia et al.

Agent57 – Ablations

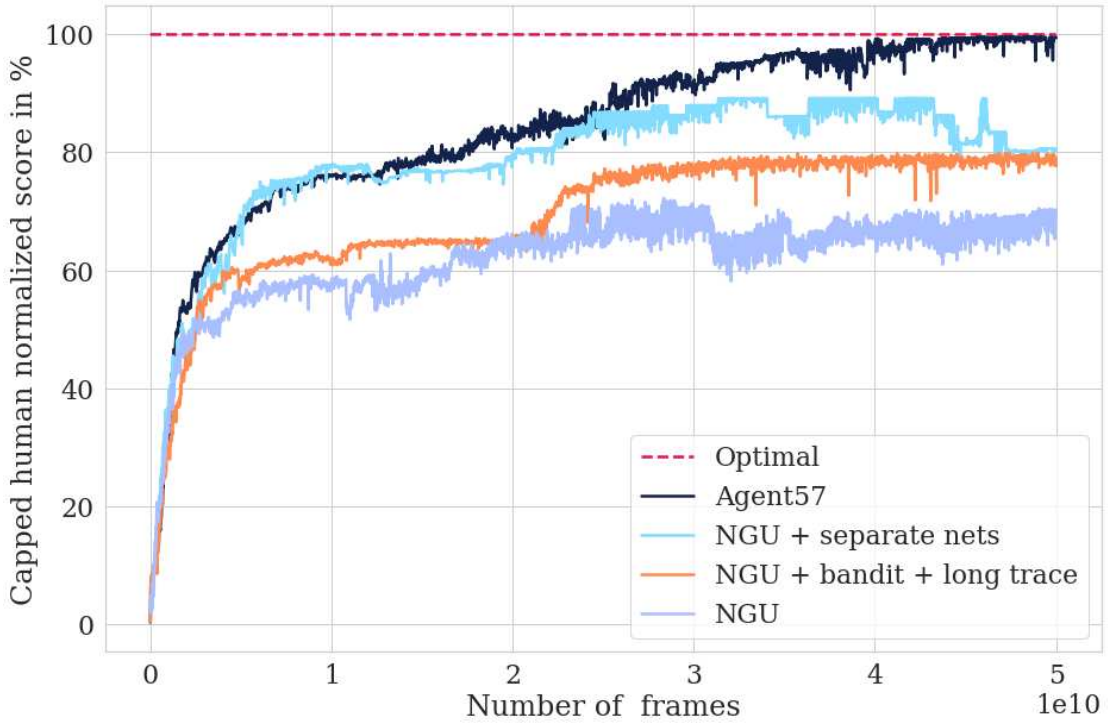


Figure 4. Performance progression on the 10-game *challenging set* obtained from incorporating each one of the improvements.

Figure 4 of "Agent57: Outperforming the Atari Human Benchmark" by A. P. Badia et al.

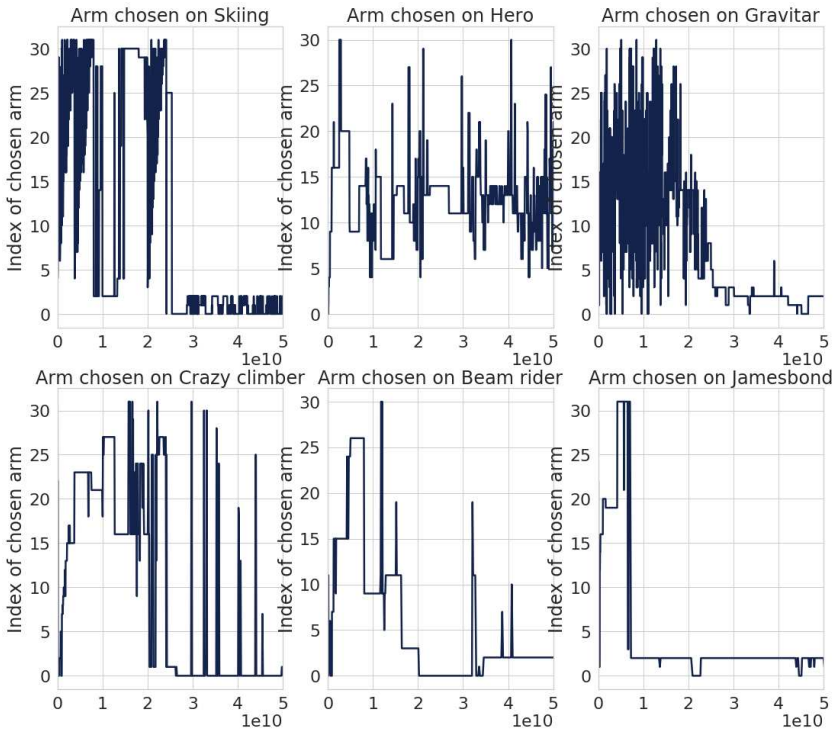


Figure 8. Best arm chosen by the evaluator of Agent57 over training for different games.

Figure 8 of "Agent57: Outperforming the Atari Human Benchmark" by A. P. Badia et al.