Importance Weighted **A**ctor-**L**earner **A**rchitecture (**IMPALA**)

arXiv '18

IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures

Lasse Espeholt [*1] Hubert Soyer [*1] Remi Munos [*1] Karen Simonyan [1] Volodymyr Mnih [1] Tom Ward [1]
Yotam Doron [1] Vlad Firoiu [1] Tim Harley [1] Iain Dunning [1] Shane Legg [1] Koray Kavukcuoglu [1]

01 Intro

Value Based RL

$$V_\theta(s) \approx V^\pi(s)$$
$$Q_\theta(s,a) \approx Q^\pi(s,a)$$

➔

Policy Based RL

$$\pi_\theta(s,a) = \mathbb{P}\left[a \mid s, \theta\right]$$

https://www.davidsilver.uk/wp-content/uploads/2020/03/pg.pdf

01

Intro

Value Based RL

$$V_\theta(s) \approx V^\pi(s)$$
$$Q_\theta(s,a) \approx Q^\pi(s,a)$$

➔

Policy Based RL

$$\pi_\theta(s,a) = \mathbb{P}\left[a \mid s, \theta\right]$$

1. Better convergence properties
2. Effective in high-dimensional or continuous action spaces
3. Can learn stochastic policies

But,

1. Typically converge to a local rather than global optimum
2. Evaluating a policy is typically inefficient and high variance

https://www.davidsilver.uk/wp-content/uploads/2020/03/pg.pdf

01

Intro

Value Based RL

$$V_\theta(s) \approx V^\pi(s)$$
$$Q_\theta(s,a) \approx Q^\pi(s,a)$$

➔

Policy Based RL

$$\pi_\theta(s,a) = \mathbb{P}[a \mid s, \theta]$$

1. Better convergence properties
2. Effective in high-dimensional or continuous action spaces
3. Can learn stochastic policies

But,

1. Typically converge to a local rather than global optimum
2. Evaluating a policy is typically inefficient and high variance

➔

Actor-Critic (& Advantage)

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)\right]$$
$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)$$

Critic Updates action-value function parameters w

Actor Updates policy parameters θ, in direction suggested by critic

$$A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$
$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ A^{\pi_\theta}(s,a)\right]$$

https://www.davidsilver.uk/wp-content/uploads/2020/03/pg.pdf

01 Intro

Value Based RL		Policy Based RL		Actor-Critic (& Advantage)
$V_\theta(s) \approx V^\pi(s)$ $Q_\theta(s,a) \approx Q^\pi(s,a)$	➔	$\pi_\theta(s,a) = \mathbb{P}[a \mid s, \theta]$	➔	$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)]$ $\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)$

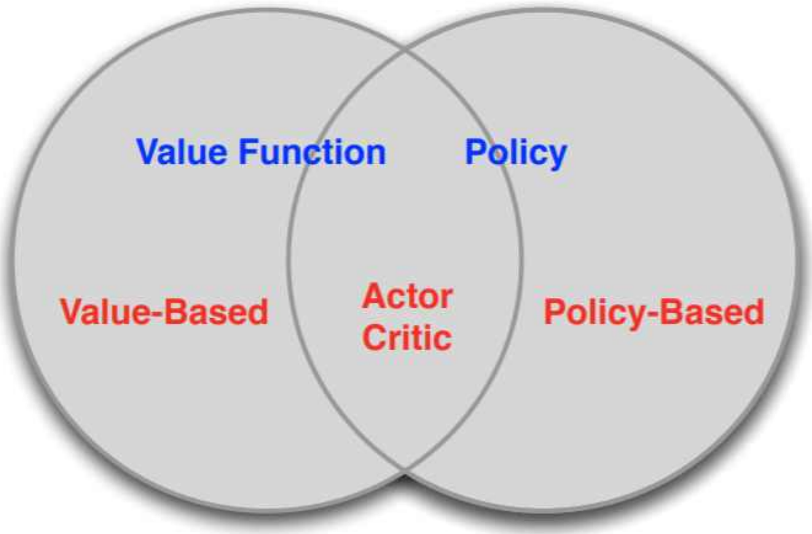


Image: https://www.davidsilver.uk/wp-content/uploads/2020/03/pg.pdf

01

Intro

Value Based RL

$$V_\theta(s) \approx V^\pi(s)$$
$$Q_\theta(s,a) \approx Q^\pi(s,a)$$

➔

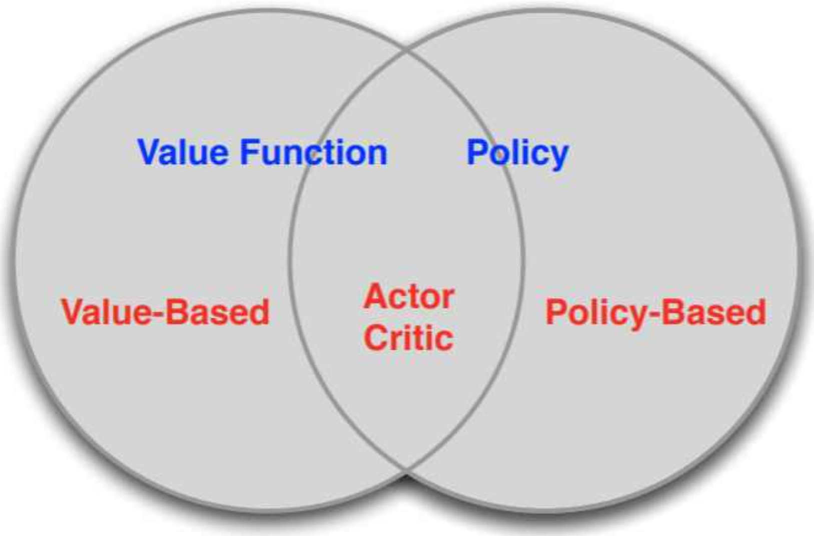
Policy Based RL

$$\pi_\theta(s,a) = \mathbb{P}\left[a \mid s, \theta\right]$$

➔

Actor-Critic (& Advantage)

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)\right]$$
$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)$$



* A3C (**A**synchronous **A**dvantage **A**ctor-Critic) - Deepmind

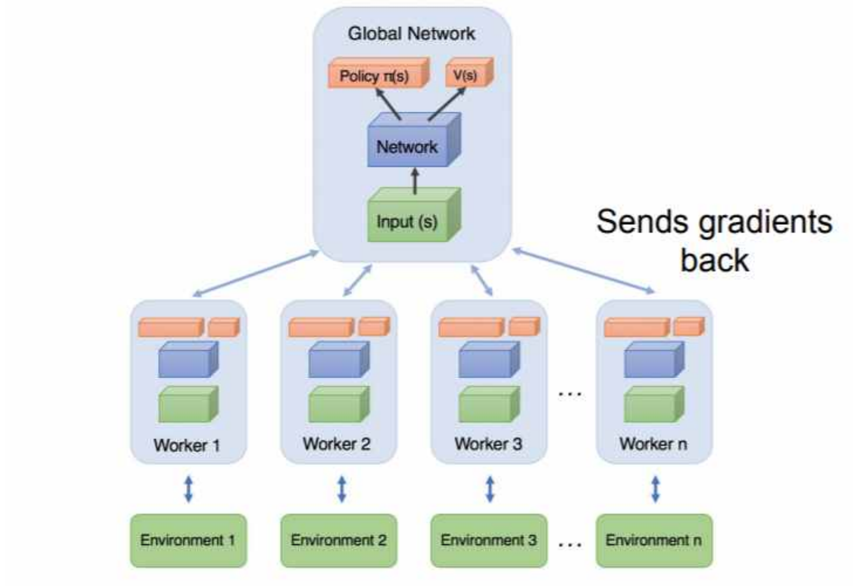
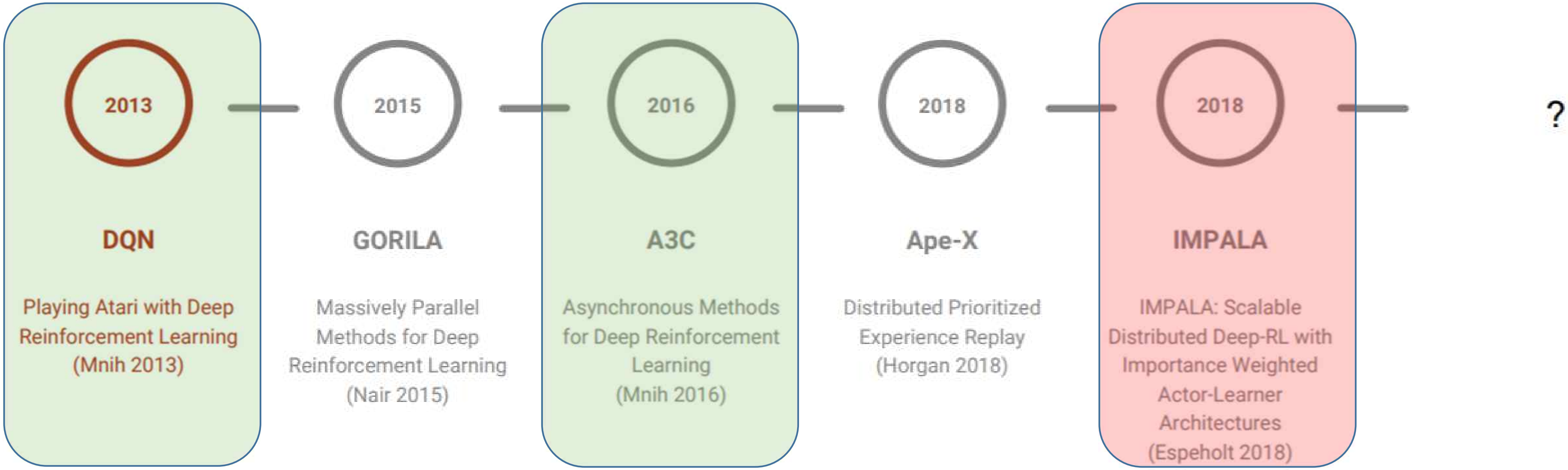


Image: http://rail.eecs.berkeley.edu/deeprlcourse-fa18/static/slides/lec-21.pdf

01

Intro



http://rail.eecs.berkeley.edu/deeprlcourse-fa18/static/slides/lec-21.pdf

01 Intro

IMPALA

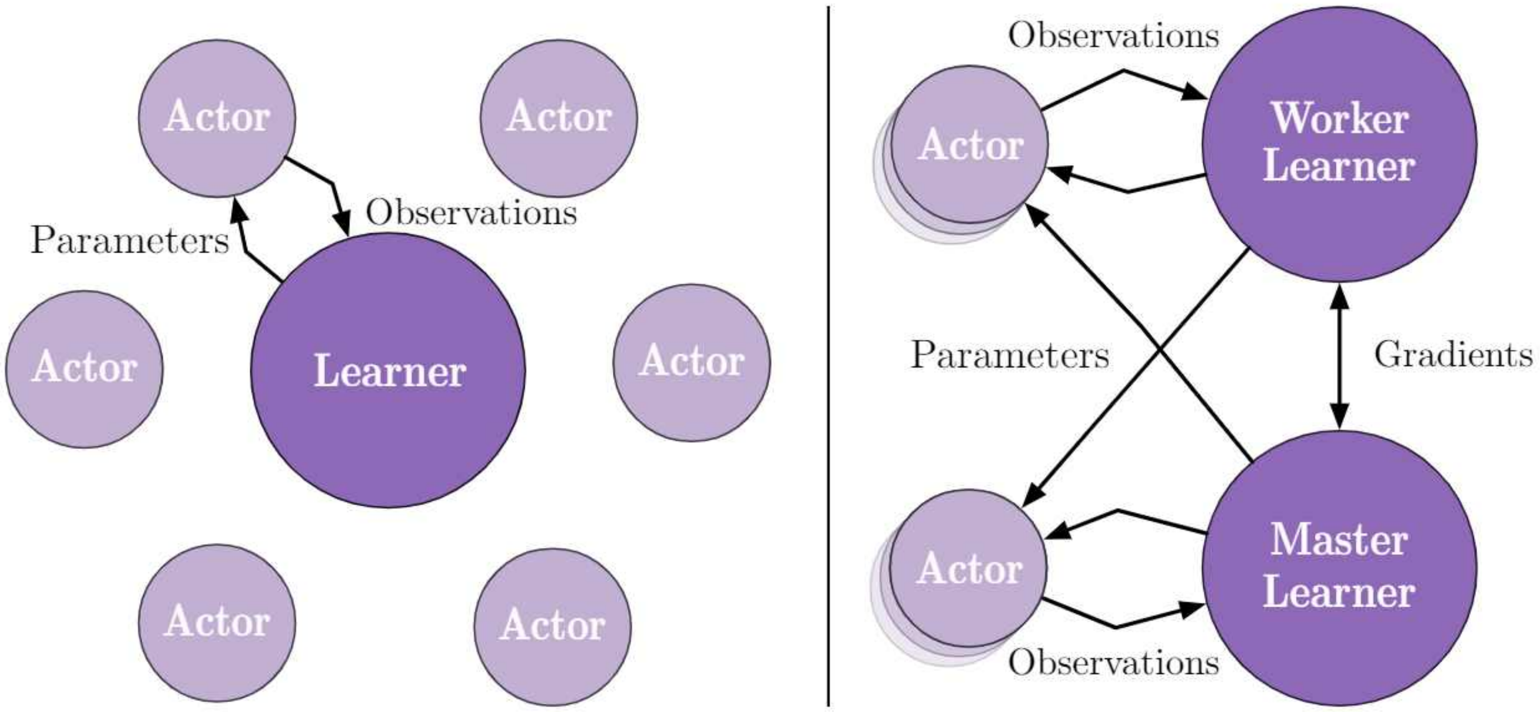
1. Efficient (in single machine)
2. Scalable (in multiple machines)
3. Stable Learning (off-policy correction & V-trace)

Vs A3C (on-policy)?

A3C: agents communicate gradients with respect to the parameters of the policy to a central parameter server

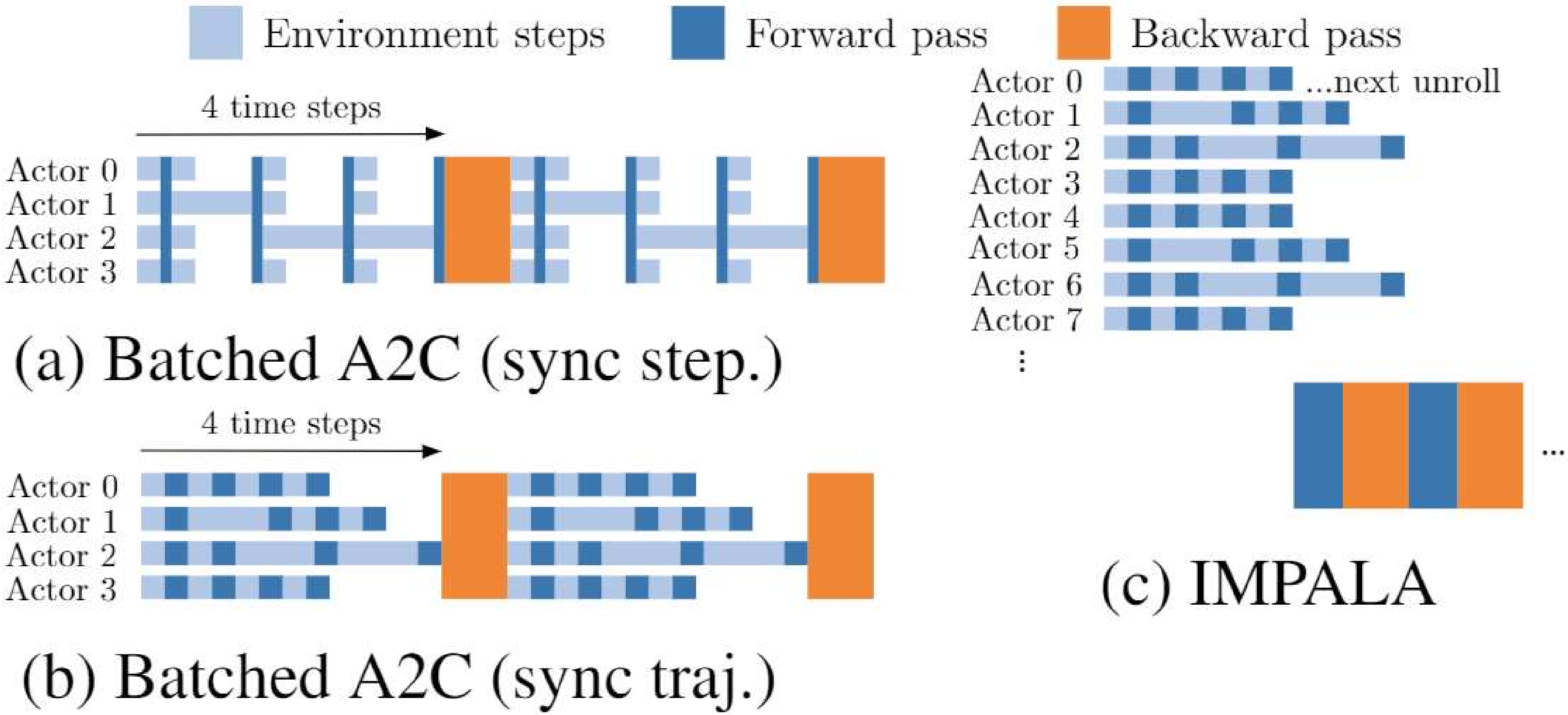
IMPALA: actors communicate trajectories of experience (s, a, r) to a centralized learner

Methods (Overview)



02

Methods (Overview)



02

Methods (Overview)

1. Beginning of each trajectory
2. Actor updates its own local policy μ to latest learner policy π
3. Runs it for n steps in its environment
4. After n steps
5. the actor sends the trajectory of s, a, r together with the corresponding policy distributions μ(a|s) and initial LSTM state to the learner through a queue
6. The learner continuously updates its policy π on batches of trajectories
7. At the time of update, policy π is potentially several updates ahead of the policy μ
   - ➔ policy-lag between the actors and leaner(s)
   - ➔ V-trace corrects for this lag

02

Methods (Detail)

Goal of an off-policy RL algorithm

1. Generating trajectories by behaviour policy μ
2. Learning the value function of target policy π

Importance Sampling

Estimate the expectation of a different distribution

$$\begin{aligned}\mathbb{E}_{X\sim P}[f(X)] &= \sum P(X)f(X) \\ &= \sum Q(X)\frac{P(X)}{Q(X)}f(X) \\ &= \mathbb{E}_{X\sim Q}\left[\frac{P(X)}{Q(X)}f(X)\right]\end{aligned}$$

https://www.davidsilver.uk/wp-content/uploads/2020/03/control.pdf

02

Methods (Detail)

7. At the time of update, policy π is potentially several updates ahead of the policy μ

➔ policy-lag between the actors and leaner(s)

➔ V-trace corrects for this lag

n-steps V-trace target

$$v_s \stackrel{\text{def}}{=} V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \Big(\prod_{i=s}^{t-1} c_i \Big) \delta_t V, \quad (1)$$

Temporal difference for V: $\delta_t V \stackrel{\text{def}}{=} \rho_t \big(r_t + \gamma V(x_{t+1}) - V(x_t) \big)$

Importance sampling weights: $\rho_t \stackrel{\text{def}}{=} \min \left(\bar{\rho}, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)} \right)$

$$c_i \stackrel{\text{def}}{=} \min \left(\bar{c}, \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} \right)$$

02

Methods (Detail)

n-steps V-trace target

$$v_s \stackrel{\text{def}}{=} V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \Big(\prod_{i=s}^{t-1} c_i \Big) \delta_t V, \quad (1)$$

Temporal difference for V: $\delta_t V \stackrel{\text{def}}{=} \rho_t \big(r_t + \gamma V(x_{t+1}) - V(x_t) \big)$

Importance sampling weights: $\rho_t \stackrel{\text{def}}{=} \min \Big(\bar{\rho}, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)} \Big)$

$$c_i \stackrel{\text{def}}{=} \min \Big(\bar{c}, \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} \Big)$$

If on-policy? ➔ π=μ

$$\begin{aligned} v_s &= V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \big(r_t + \gamma V(x_{t+1}) - V(x_t) \big) \\ &= \sum_{t=s}^{s+n-1} \gamma^{t-s} r_t + \gamma^n V(x_{s+n}), \qquad (2) \end{aligned}$$

02

Methods (Detail)

n-steps V-trace target

$$v_s \stackrel{\text{def}}{=} V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \Big(\prod_{i=s}^{t-1} c_i \Big) \delta_t V, \quad (1)$$

Temporal difference for V: $\delta_t V \stackrel{\text{def}}{=} \rho_t \big(r_t + \gamma V(x_{t+1}) - V(x_t) \big)$

Importance sampling weights: $\rho_t \stackrel{\text{def}}{=} \min \big(\bar{\rho}, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)} \big)$

$c_i \stackrel{\text{def}}{=} \min \big(\bar{c}, \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} \big)$

If on-policy? ➔ π=μ

$$\begin{aligned} v_s &= V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \big(r_t + \gamma V(x_{t+1}) - V(x_t) \big) \\ &= \sum_{t=s}^{s+n-1} \gamma^{t-s} r_t + \gamma^n V(x_{s+n}), \qquad (2) \end{aligned}$$

➔ On-policy n-step Bellman target !

02 Methods (Detail)

n-steps V-trace target

$$v_s \stackrel{\text{def}}{=} V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \Big(\prod_{i=s}^{t-1} c_i \Big) \delta_t V, \quad (1)$$

Temporal difference for V: $\delta_t V \stackrel{\text{def}}{=} \rho_t \big(r_t + \gamma V(x_{t+1}) - V(x_t) \big)$

Importance sampling weights: $\rho_t \stackrel{\text{def}}{=} \min \big(\bar{\rho}, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)} \big)$

$$c_i \stackrel{\text{def}}{=} \min \big(\bar{c}, \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} \big)$$

If on-policy? ➔ π=μ

$$\begin{aligned} v_s &= V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \big(r_t + \gamma V(x_{t+1}) - V(x_t) \big) \\ &= \sum_{t=s}^{s+n-1} \gamma^{t-s} r_t + \gamma^n V(x_{s+n}), \qquad (2) \end{aligned}$$

➔ On-policy n-step Bellman target !

➔ We can use the same algorithm for off- and on-policy data !

02

Methods (Detail)

n-steps V-trace target

$$v_s \stackrel{\text{def}}{=} V(x_s) + \underbrace{\sum_{t=s}^{s+n-1} \gamma^{t-s} \Big(\prod_{i=s}^{t-1} c_i \Big) \delta_t V}_{\text{Correction}}$$

Temporal difference for V: $\delta_t V \stackrel{\text{def}}{=} \rho_t \big(r_t + \gamma V(x_{t+1}) - V(x_t) \big)$

Importance sampling weights: $\rho_t \stackrel{\text{def}}{=} \min \left(\bar{\rho}, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)} \right)$

$$c_i \stackrel{\text{def}}{=} \min \left(\bar{c}, \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} \right)$$

02 Methods (Detail)

n-steps V-trace target

$$v_s \stackrel{\text{def}}{=} V(x_s) + \underbrace{\sum_{t=s}^{s+n-1} \gamma^{t-s} \Big(\prod_{i=s}^{t-1} c_i \Big) \delta_t V}_{\text{Correction}}$$

Temporal difference for V: $\delta_t V \stackrel{\text{def}}{=} \rho_t \big(r_t + \gamma V(x_{t+1}) - V(x_t) \big)$

Importance sampling weights: $\rho_t \stackrel{\text{def}}{=} \min\left(\bar{\rho}, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}\right)$ ← Which value function μ or π

Appendix A

$c_i \stackrel{\text{def}}{=} \min\left(\bar{c}, \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)}\right)$ ← Speed of convergence

ρ defines the fixed point of update rule !

If ρ is close to 1?
If ρ is close to zero?

c used as a variance reduction technique !

Notice that this does not impact the solution to which we converge.

02 Methods (Detail)

Actor–Critic algorithm (Policy Gradient)

On-policy case

Gradient of the value function:

$$\nabla V^{\mu}(x_0) = \mathbb{E}_{\mu}\Big[\sum_{s\geq 0}\gamma^{s}\nabla \log \mu(a_s|x_s)Q^{\mu}(x_s,a_s)\Big],$$

$$Q^{\mu}(x_s,a_s) \stackrel{\text{def}}{=} \mathbb{E}_{\mu}\big[\sum_{t>s}\gamma^{t-s}r_t|x_s,a_s\big]$$

Update the policy parameters in the direction of

$$\mathbb{E}_{a_s\sim\mu(\cdot|x_s)}\Big[\nabla \log \mu(a_s|x_s)q_s\Big|x_s\Big]$$

Off-policy case

Appendix A & E.3

$$\mathbb{E}_{a_s\sim\mu(\cdot|x_s)}\left[\frac{\pi_{\bar{\rho}}(a_s|x_s)}{\mu(a_s|x_s)}\nabla \log \pi_{\bar{\rho}}(a_s|x_s)q_s\Big|x_s\right]$$

V-trace estimate

$$q_s \stackrel{\text{def}}{=} r_s + \gamma \underline{v_{s+1}}$$

02

Methods (Detail)

Canonical V-Trace Actor–Critic algorithm (Policy Gradient)

Appendix A & E.3

$$\mathbb{E}_{a_s \sim \mu(\cdot|x_s)} \left[\frac{\pi_{\bar{\rho}}(a_s|x_s)}{\mu(a_s|x_s)} \nabla \log \pi_{\bar{\rho}}(a_s|x_s) q_s \Big| x_s \right]$$

$\Downarrow$

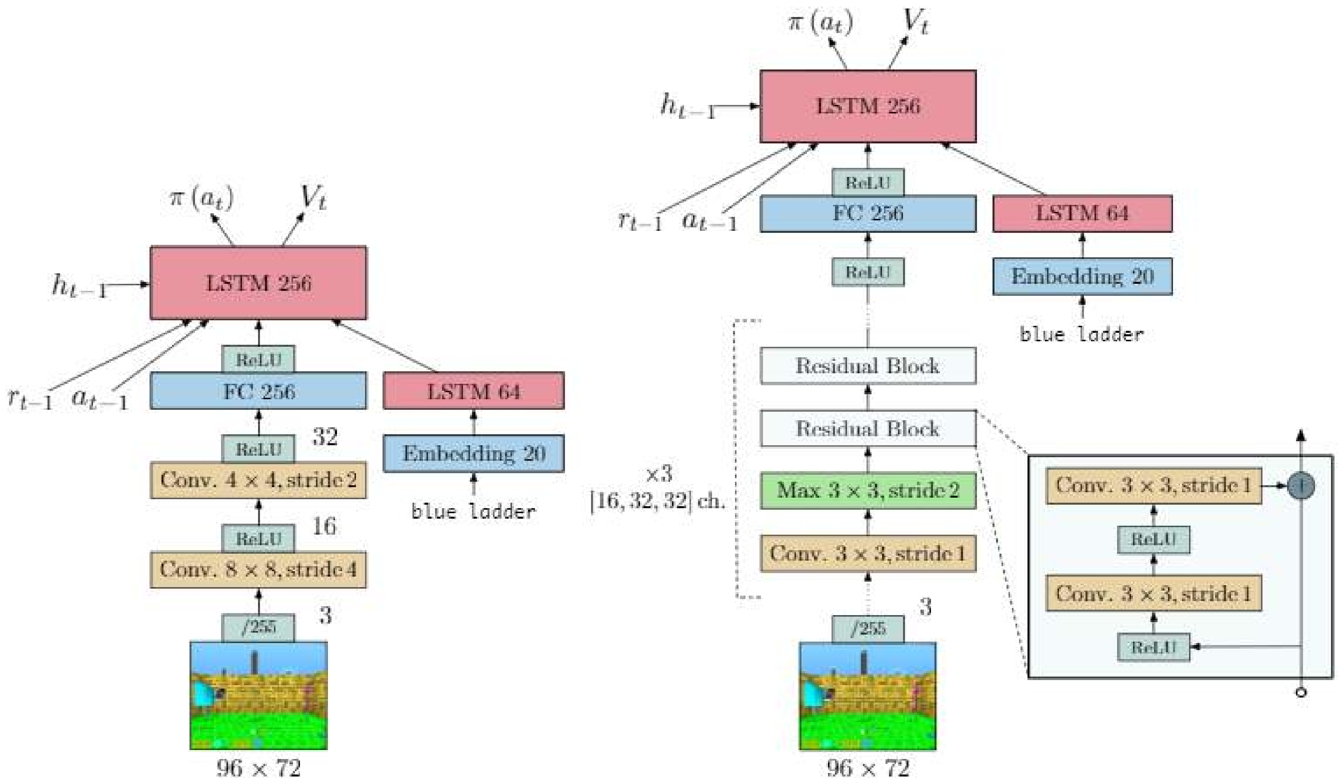
$$\rho_s \nabla_\omega \log \pi_\omega(a_s|x_s) \big(r_s + \gamma v_{s+1} - V_\theta(x_s) \big) \quad \underline{-\nabla_\omega \sum_a \pi_\omega(a|x_s) \log \pi_\omega(a|x_s)}$$

Prevent premature convergence:
Adding entropy bonus, like in A3C

03

Experiment

Model architecture



03

Experiment

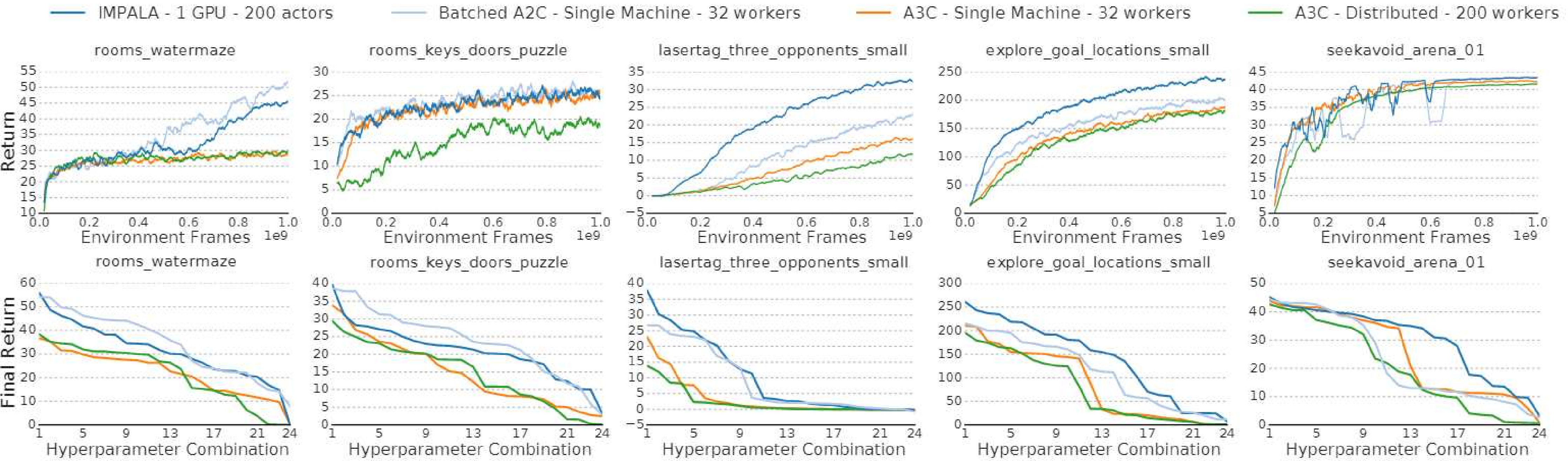
Model architecture

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.

Hyperparameter:
Appendix D.1

03 Experiment



03

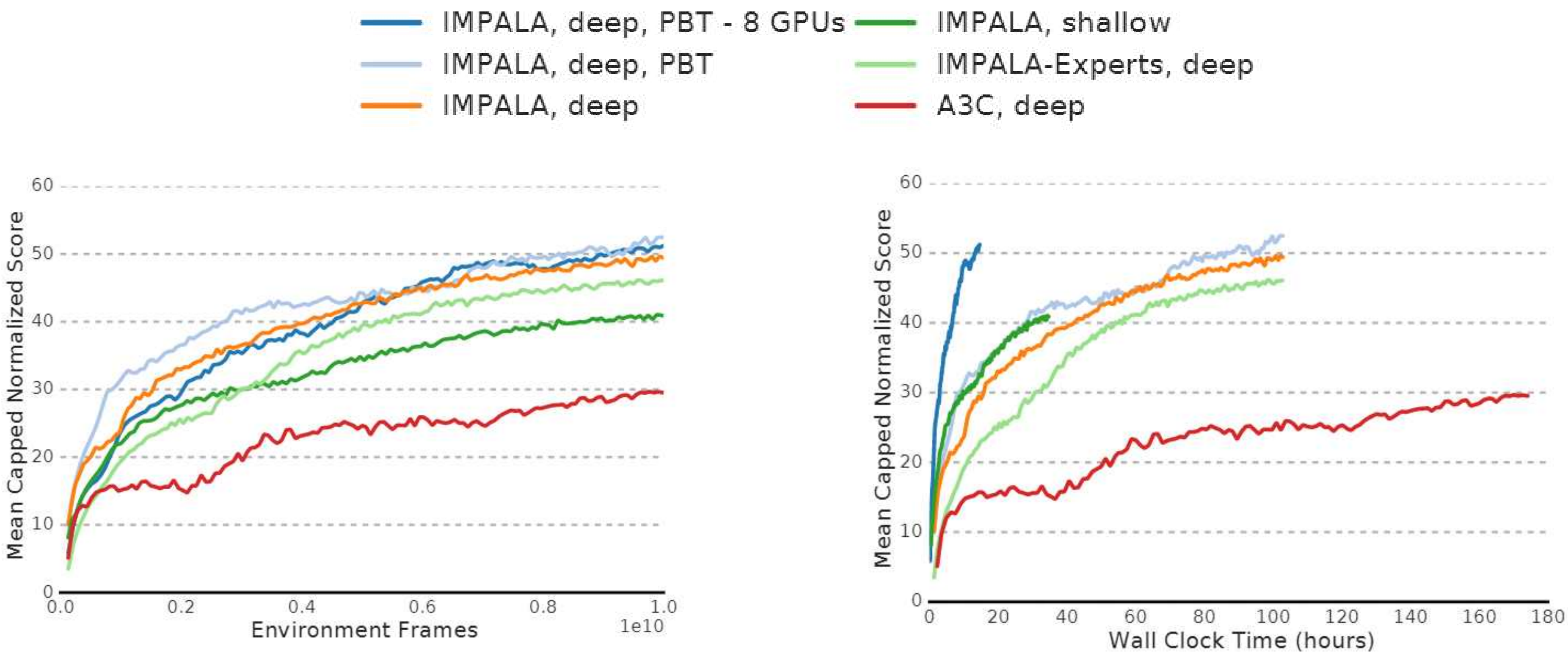
Experiment

	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`

03

Experiment



04

Conclusion

IMPALA

➔ A new highly scalable distributed agent, and a new off-policy learning algorithm, V-trace.

V-trace

➔ A general off-policy learning algorithm that is more stable and robust for actor critic agents.

Experiments

➔ IMPALA is the first Deep-RL agent that has been successfully tested in such large-scale multi-task settings and it has shown superior performance compared to A3C based agents.

Q&A