Efficiency and Imitation Learning

Xuehai Pan

Center on Frontiers of Computing Studies
Peking University

June 2nd, 2020

Outline

Sample Efficient Actor-Critic with Experience Replay

Ziyu Wang[1] Victor Bapst[1] Nicolas Heess[1] Volodymyr Mnih[1]

Remi Munos[1] Koray Kavukcuoglu[1] Nando de Freitas[1 2 3]

[1]DeepMind [2]CIFAR [3]Oxford University

ICLR 2017

Motivation

- Environment simulation cost is expensive.
- Deep Q-learning methods:
  - deterministic policy is limited in adversarial domains.
  - greedy action search is costly for large action spaces.
- Policy gradient methods are sample inefficient.

Problem Setup

observation: $x_t \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$

action: $a_t \in \mathcal{A}$ (discrete: $\mathcal{A} = \{1, 2, \ldots, N_a\}$; continuous: $\mathcal{A} \subseteq \mathbb{R}^{n_a}$)

reward: $r_t \in \mathbb{R}$

policy: $\pi(a|x_t)$

goal:

$$\max_{\pi} R_t = \sum_{i \geq 0} \gamma^i r_{t+i}, \tag{1.1}$$

functions:

$$\begin{aligned} Q^{\pi}(x_t, a_t) &= \mathbb{E}_{x_{t+1:\infty}, a_{t+1:\infty}}[R_t | x_t, a_t], \\ V^{\pi}(x_t) &= \mathbb{E}_{a_t}[Q^{\pi}(x_t, a_t) | x_t], \\ A^{\pi}(x_t, a_t) &= Q^{\pi}(x_t, a_t) - V^{\pi}(x_t). \end{aligned} \tag{1.2}$$

gradient of policy:

- AC:

$$
\begin{aligned}
g &= \mathbb{E}_{x_{0:\infty}, a_{0:\infty}} \left[\sum_{t \geq 0} A^{\pi}(x_t, a_t) \nabla_{\theta} \log \pi_{\theta}(a_t | x_t) \right] \\
&= \mathbb{E}_{x_{0:\infty}, a_{0:\infty}} \left[\sum_{t \geq 0} (r_t + \gamma V^{\pi}(x_{t+1}) - V^{\pi}(x_t)) \nabla_{\theta} \log \pi_{\theta}(a_t | x_t) \right],
\end{aligned}
\tag{1.3}
$$

- A3C:

$$
\hat{g}^{\mathrm{a3c}} = \sum_{t \geq 0} \left[\left(\left(\sum_{i=0}^{k-1} \gamma^i r_{t+i} \right) + \gamma^k V_{\theta_v}^{\pi}(x_{t+k}) - V_{\theta_v}^{\pi}(x_t) \right) \nabla_{\theta} \log \pi_{\theta}(a_t | x_t) \right].
\tag{1.4}
$$

Outline

Importance Sampling

behavior policy: $\mu(a|x)$

trajectory: $\{x_0, a_0, r_0, \mu(\cdot|x_0), \ldots, x_k, a_k, r_k, \mu(\cdot|x_k)\}$

Importance weighted policy gradient:

$$\hat{g}^{\text{imp}} = \left(\prod_{t=0}^{k} \rho_t\right) \sum_{t=0}^{k} \left[\left(\sum_{i=0}^{k} \gamma^i r_{t+i}\right) \nabla_\theta \log \pi_\theta(a_t|x_t)\right], \tag{1.5}$$

where $\rho_t = \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}$ denotes the importance weight.

- importance sampling: unbiased but has very high variance.
- truncated importance sampling: bounded variance but introduces bias.

Marginal Value Function Approximation

Policy gradient approximation using marginal value functions:

$$g^{\text{marg}} = \mathbb{E}_{x_t \sim \beta, a_t \sim \mu}[\rho_t \nabla_\theta \log \pi_\theta(a_t|x_t) Q^\pi(x_t, a_t)], \tag{1.6}$$

where $\mathbb{E}_{x_t \sim \beta, a_t \sim \mu}[\cdot]$ is the expectation with respect to the limiting distribution $\beta(x) = \lim_{t \to \infty} P(x_t = x | x_0, \mu)$ with behavior policy μ.

- gradient depends on Q^π and not on Q^μ,
- no longer have a product of importance weights.

Retrace (λ)

The general operator that we consider for comparing several return-based off-policy algorithms is:

$$\mathcal{R}Q(x,a) = Q(x,a) + \mathbb{E}_\mu \left[\sum_{t \geq 0} \gamma^t \left(\sum_{s=1}^{t} c_s \right) \left(r_t + \gamma \mathbb{E}_\pi [Q(x,\cdot)] - Q(x_t, a_t) \right) \right], \tag{1.7}$$

where $\mathbb{E}_\pi[Q(x,\cdot)] = \sum_a \pi(a|x) Q(x,a)$.

	Definition of c_s	Estimation variance	Guaranteed convergence	Use full returns (near on-policy)
Importance sampling	$\frac{\pi(a_t \mid x_s)}{\mu(a_t \mid x_s)}$	High	for any π, μ	yes
$Q^\pi(\lambda)$	λ	Low	for π close to μ	yes
$\mathrm{TB}(\lambda)$	$\lambda \pi(a_t \mid x_s)$	Low	for any π, μ	no
$\mathrm{Retrace}(\lambda)$	$\lambda \min\left\{1, \frac{\pi(a_t \mid x_s)}{\mu(a_t \mid x_s)}\right\}$	Low	for any π, μ	yes

Action-value function estimation using Retrace ($\lambda = 1$):

$$Q^{\text{ret}}(x_t, a_t) = r_t + \gamma\bar{\rho}_{t+1}\left[Q^{\text{ret}}(x_{t+1}, a_{t+1}) - Q(x_{t+1}, a_{t+1})\right] + \gamma V(x_{t+1}), \tag{1.8}$$

where $\bar{\rho}_t$ is the truncated importance weight, $\bar{\rho}_t = \min\{c, \rho_t\}$, Q is the current value estimate of Q^π, and $V(x) = \mathbb{E}_{a\sim\pi}[Q(x, a)]$.

ACER uses Q^{ret} to estimate Q^π, which can significantly reduce bias in the estimation of the policy gradient.

Importance Weight Truncation

with Bias Correction

Truncated importance weight with correction term:

$$
\begin{aligned}
g^{\text{marg}} =& \mathbb{E}_{\substack{x_t\sim\beta \\ a_t\sim\mu}}[\rho_t \nabla_\theta \log \pi_\theta(a_t|x_t) Q^\pi(x_t, a_t)] \\
=& \quad \mathbb{E}_{\substack{x_t\sim\beta \\ a_t\sim\mu}}[\bar{\rho}_t \nabla_\theta \log \pi_\theta(a_t|x_t) Q^\pi(x_t, a_t)] \\
&+ \mathbb{E}_{\substack{x_t\sim\beta \\ a\sim\pi}}\left[\left[\frac{\rho_t(a) - c}{\rho_t(a)}\right]_+ \nabla_\theta \log \pi_\theta(a|x_t) Q^\pi(x_t, a)\right]
\end{aligned}
\tag{1.9}
$$

where $\bar{\rho}_t = \min\{c, \rho_t\}$ with $\rho_t = \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}$, and $\rho_t(a) = \frac{\pi(a|x_t)}{\mu(a|x_t)}$, and $[x]_+ = x$ if $x > 0$ and it is zero otherwise.

Importance Weight Truncation

with Bias Correction

Use neural network $Q_{\theta_v}(x, a)$ to approximate $Q^{\pi}(x, a)$:

$$\begin{aligned}\hat{g}^{\text{marg}} = \quad & \mathbb{E}_{\substack{x_t \sim \beta \\ a_t \sim \mu}}\left[\bar{\rho}_t \nabla_\theta \log \pi_\theta(a_t|x_t) Q^{\text{ret}}(x_t, a_t)\right] \\ & + \mathbb{E}_{\substack{x_t \sim \beta \\ a \sim \pi}}\left[\left[\frac{\rho_t(a) - c}{\rho_t(a)}\right]_+ \nabla_\theta \log \pi_\theta(a|x_t) Q_{\theta_v}(x_t, a)\right]\end{aligned} \tag{1.10}$$

Off-policy ACER gradient:

$$\begin{aligned}\hat{g}_t^{\text{acer}} = \quad & \bar{\rho}_t \nabla_\theta \log \pi_\theta(a_t|x_t)\left[Q^{\text{ret}}(x_t, a_t) - V_{\theta_v}(x_t)\right] \\ & + \mathbb{E}_{a \sim \pi}\left[\left[\frac{\rho_t(a) - c}{\rho_t(a)}\right]_+ \nabla_\theta \log \pi_\theta(a|x_t)\left[Q_{\theta_v}(x_t, a) - V_{\theta_v}(x_t)\right]\right]\end{aligned} \tag{1.11}$$

- $c \to \infty$: off-policy policy gradient using Retrace.
- $c = 0$: actor critic that depends entirely on Q estimates.

Efficient Trust Region Policy Optimization

Decompose the policy network $\pi(\cdot|x)$ in two parts: a distribution f, and a neural network that generates the statistics $\phi_\theta(x)$ of this distribution. That is, given f, the policy is completely characterized by the network ϕ_θ: $\pi(\cdot|x) = f(\cdot|\phi_\theta(x))$.

Off-policy ACER gradient with respect to ϕ:

$$\begin{aligned}\hat{g}_t^{\text{acer}} = \quad & \bar{\rho}_t \nabla_{\phi_\theta(x_t)} \log f(a_t|\phi_\theta(x_t)) \left[Q^{\text{ret}}(x_t, a_t) - V_{\theta_v}(x_t)\right] \\ & + \mathbb{E}_{a\sim\pi}\left[\left[\frac{\rho_t(a) - c}{\rho_t(a)}\right]_+ \nabla_{\phi_\theta(x_t)} \log f(a|\phi_\theta(x_t)) [Q_{\theta_v}(x_t, a) - V_{\theta_v}(x_t)]\right]\end{aligned} \tag{1.12}$$

First stage of optimization

Solve the following optimization problem with a linearized KL divergence constraint:

$$\begin{aligned} &\min_z \frac{1}{2}\|\hat{g}_t^{\text{acer}} - z\|_2^2, \\ &\text{s.t. } k^T z \leq \delta, \\ &\quad k = \nabla_{\phi_\theta(x_t)} D_{KL}(f(\cdot|\phi_{\theta_a}(x_t)) \| f(\cdot|\phi_\theta(x_t))), \end{aligned} \tag{1.13}$$

where ϕ_{θ_a} is the average policy network, and update its parameters θ_a "softly" after each update to the policy parameter θ:

$$\theta_a \leftarrow \alpha\theta_a + (1-\alpha)\theta. \tag{1.14}$$

The solution is:

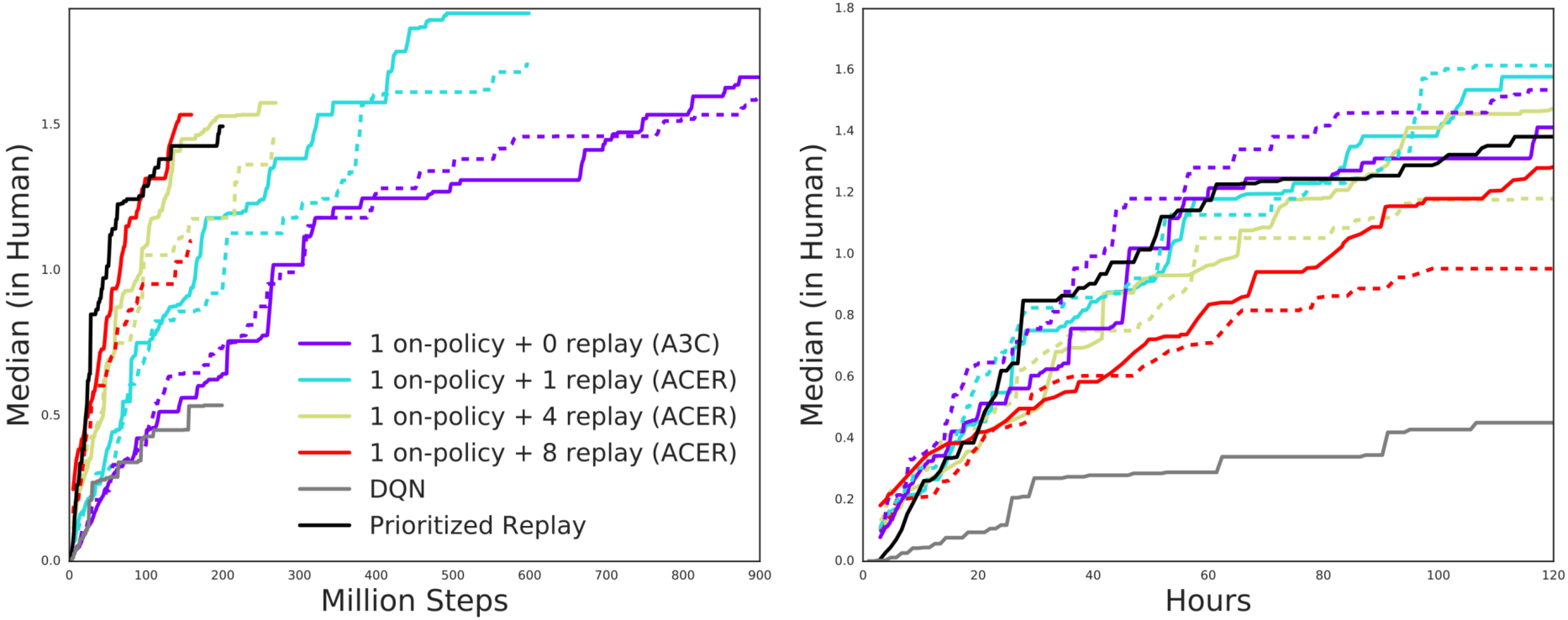
$$z^* = \hat{g}_t^{\text{acer}} - \max\left\{0, \frac{k^T \hat{g}_t^{\text{acer}} - \delta}{\|k\|_2^2}\right\} k \tag{1.15}$$

Second stage of optimization

The parameter updates for the policy network follow from the chain rule:

$$\theta \leftarrow \theta + \eta \frac{\partial \phi_\theta(x_t)}{\partial \theta} z^* \tag{1.16}$$

Results on Atari



ACER improvements in sample (**LEFT**) and computation (**RIGHT**) complexity on Atari. On each plot, the median of the human-normalized score across all 57 Atari games is presented for 4 ratios of replay with 0 replay corresponding to on-policy A3C. The colored solid and dashed lines represent ACER with and without trust region updating respectively. The environment steps are counted over all threads. The gray curve is the original DQN agent (Mnih et al., 2015) and the black curve is one of the Prioritized Double DQN agents from Schaul et al. (2016).

Outline

Policy Evaluation

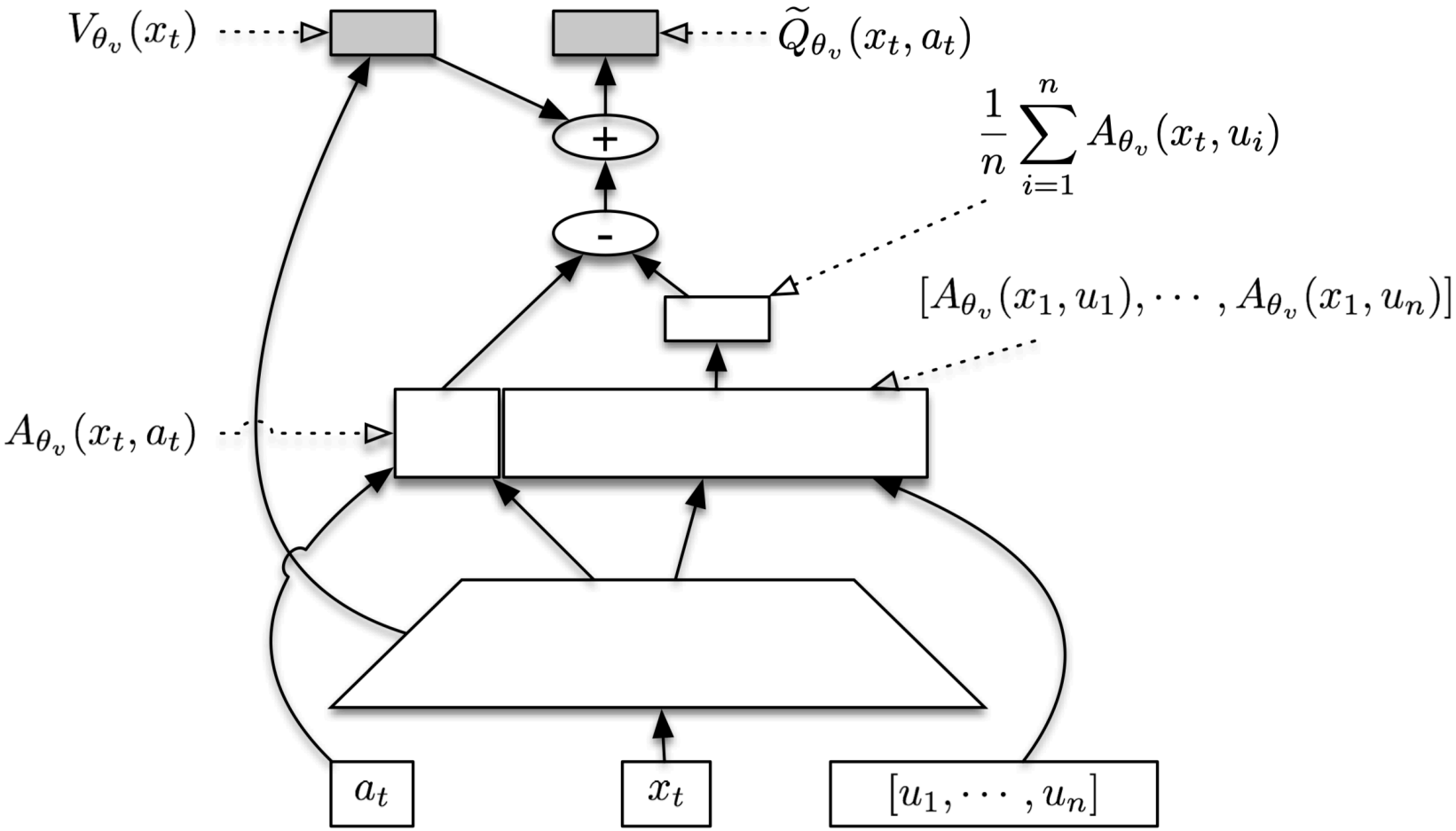
At each time step t, a Stochastic Dueling Network (SDN) outputs a stochastic estimate $\tilde{Q}_{\theta_v}$ of Q^π and a deterministic estimate V_{θ_v} of V^π, such that

$$\tilde{Q}_{\theta_v}(x_t, a_t) \sim V_{\theta_v}(x_t) + A_{\theta_v}(x_t, a_t) - \frac{1}{n}\sum_{i=1}^{n} A_{\theta_v}(x_t, u_i), \tag{1.17}$$

$$\mathbb{E}_{a\sim\pi(\cdot|x_t)}\left[\mathbb{E}_{u_{1:n}\sim\pi(\cdot|x_t)}\left[\tilde{Q}_{\theta_v}(x_t, a)\right]\right] = V_{\theta_v}(x_t). \tag{1.18}$$

where n is a parameter and $u_{1:n} \sim \pi_\theta(\cdot|x_t)$.

Stochastic Dueling Network



A schematic of the Stochastic Dueling Network. In the drawing, $[u_1, \ldots, u_n]$ are assumed to be samples from $\pi_\theta(\cdot|x_t)$. This schematic illustrates the concept of SDNs but does not reflect the real sizes of the networks used.

Target for estimating V^{π}:

$$V^{\text{target}}(x_t) = \min\left\{1, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}\right\}\left(Q^{\text{ret}}(x_t, a_t) - Q_{\theta_v}(x_t, a_t)\right) + V_{\theta_v}(x_t). \tag{1.19}$$

In implementation, the truncated importance weights:

$$\bar{\rho}_t = \min\left\{1, \left(\frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}\right)^{\frac{1}{n_a}}\right\}. \tag{1.20}$$

Although not essential, we have found this formulation to lead to faster learning.

Trust Region Updating

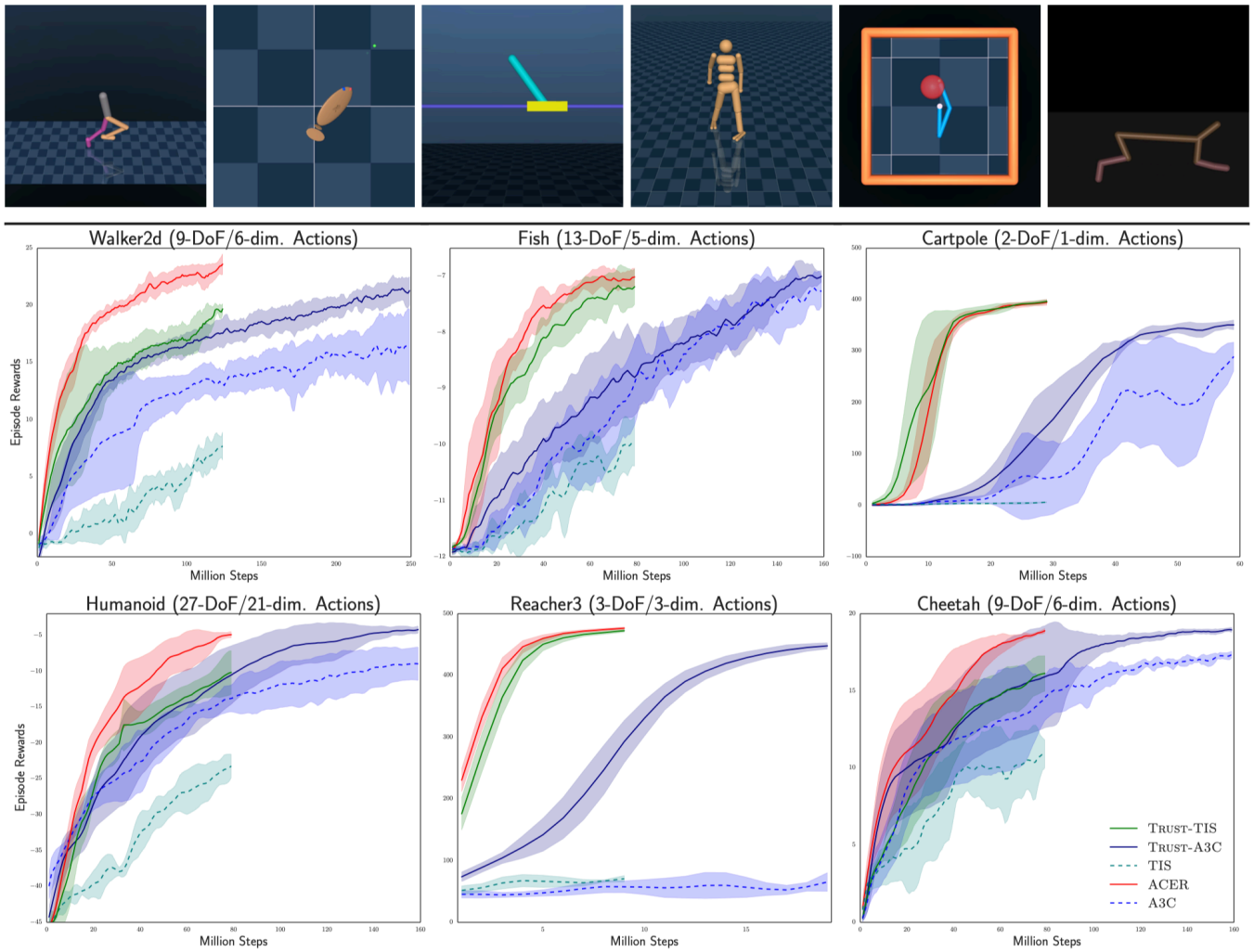
Off-policy ACER gradient with respect to ϕ:

$$
\begin{aligned}
g^{\text{acer}} = \; & \mathbb{E}_{\substack{x_t \sim \beta \\ a_t \sim \mu}} \left[\bar{\rho}_t \nabla_{\phi_\theta(x_t)} \log f(a_t | \phi_\theta(x_t)) \left[Q^{\text{ret}}(x_t, a_t) - V_{\theta_v}(x_t) \right] \right] \\
& + \mathbb{E}_{\substack{x_t \sim \beta \\ a \sim \pi}} \left[\left[\frac{\rho_t(a) - c}{\rho_t(a)} \right]_+ \nabla_{\phi_\theta(x_t)} \log f(a | \phi_\theta(x_t)) \left[\tilde{Q}_{\theta_v}(x_t, a) - V_{\theta_v}(x_t) \right] \right]
\end{aligned}
\tag{1.21}
$$

Here, $Q^{\text{opc}}(x_t, a_t)$ is the same as Retrace with the exception that the truncated importance ratio is replaced with 1.
Sample $a'_t \sim \pi_\theta(\cdot | x_t)$ to obtain the Monte Carlo approximation:

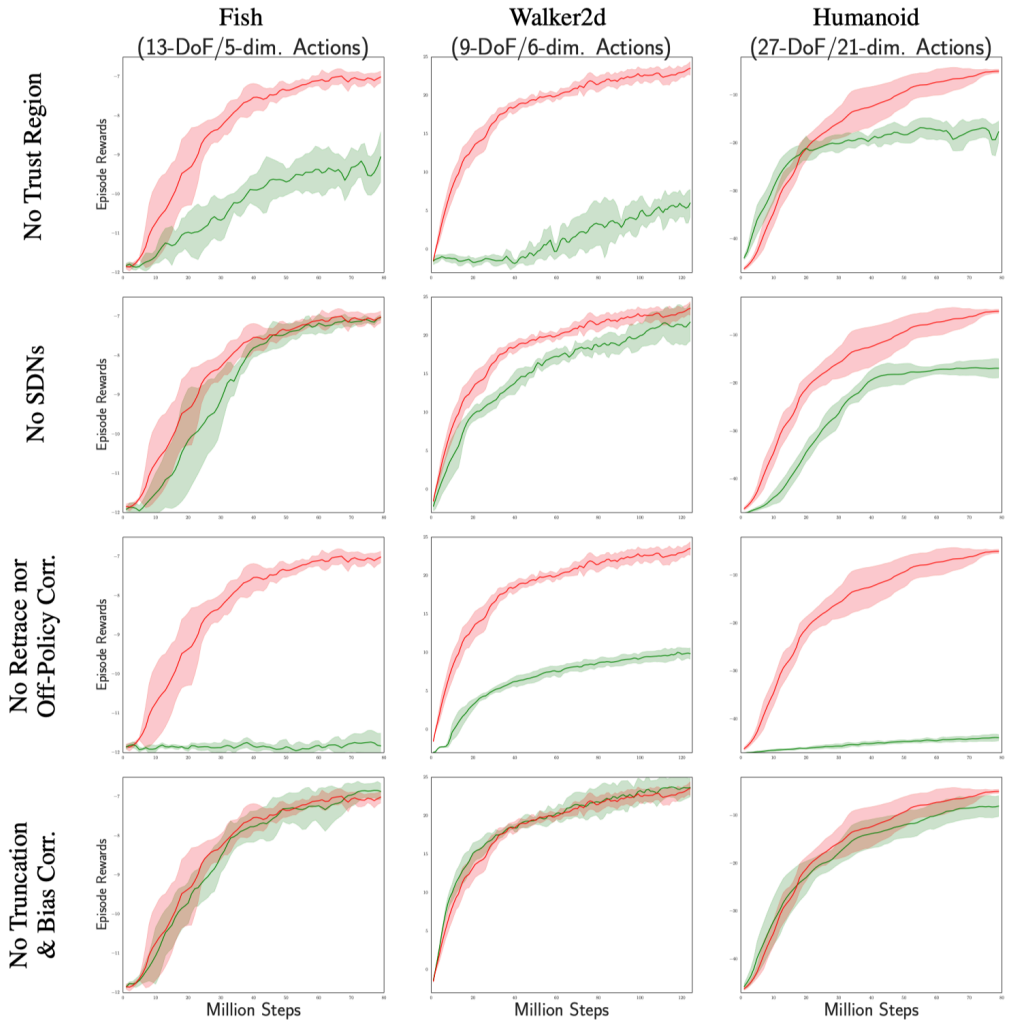
$$
\begin{aligned}
\hat{g}_t^{\text{acer}} = \; & \bar{\rho}_t \nabla_{\phi_\theta(x_t)} \log f(a_t | \phi_\theta(x_t)) [Q^{\text{opc}}(x_t, a_t) - V_{\theta_v}(x_t)] \\
& + \left[\frac{\rho_t(a'_t) - c}{\rho_t(a'_t)} \right]_+ \nabla_{\phi_\theta(x_t)} \log f\left(a'_t \middle| \phi_\theta(x_t)\right) \left[\tilde{Q}_{\theta_v}(x_t, a'_t) - V_{\theta_v}(x_t) \right]
\end{aligned}
\tag{1.22}
$$

Results on MuJoCo



[**TOP**] Screen shots of the continuous control tasks. [**BOTTOM**] Performance of different methods on these tasks. ACER outperforms all other methods and shows clear gains for the higherdimensionality tasks (humanoid, cheetah, walker and fish). The proposed trust region method by itself improves the two baselines (truncated importance sampling and A3C) significantly.

Results on MuJoCo



Ablation analysis evaluating the effect of different components of ACER. Each row compares ACER with and without one component. The columns represents three control tasks. Red lines, in all plots, represent ACER whereas green lines ACER with missing components. This study indicates that all 4 components studied improve performance where 3 are critical to success. Note that the ACER curve is of course the same in all rows.

Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion

Jacob Buckman Danijar Hafner George Tucker

Eugene Brevdo Honglak Lee

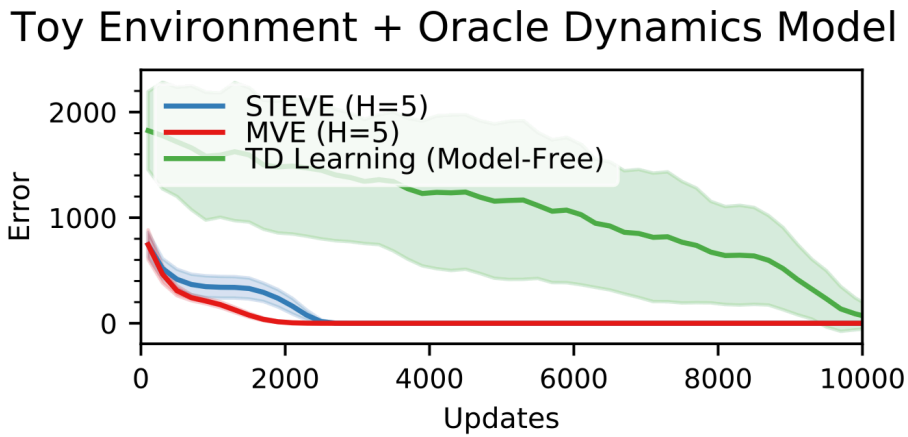
Google Brain, Mountain View, CA, USA

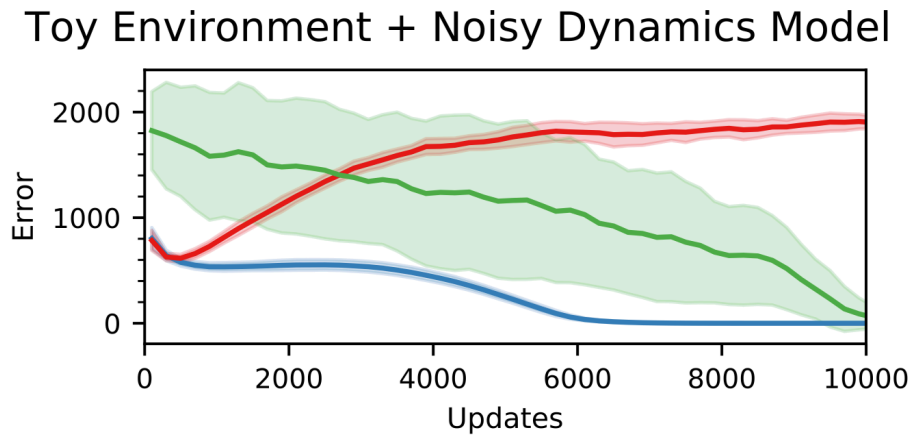
NeurIPS 2018

Motivation

- Deep model-free RL algorithms is sample inefficient.
- Model-based approaches:
  - difficult to learn an accurate model of complex environments.
  - inaccurate model can cause the wrong policy to be learned.

STochastic Ensemble Value Expansion





Value error per update on a value-estimation task (fixed policy) in a toy environment. H is the maximum rollout horizon. When given access to a perfect dynamics model, hybrid model-free model-based approaches (MVE and STEVE) solve this task with $5\times$ fewer samples than model-free TD learning. However, when only given access to a noisy dynamics model, MVE diverges due to model errors. In contrast, STEVE converges to the correct solution, and does so with a $2\times$ speedup over TD learning. This is because STEVE dynamically adapts its rollout horizon to accommodate model error.

Value Estimation with TD-learning

$Q^\pi(s,a)$ satisfies a recursion relation:

$$Q^\pi(s,a) = r(s,a) + \gamma\big(1 - d(s')\big)Q^\pi(s', \pi(s')), \tag{2.1}$$

where $s' = T(s,a)$ and $d(s')$ is an indicator function which returns 1 when s' is a terminal state and 0 otherwise.

Approximate $Q^\pi(s,a)$ with a deep neural network, $\hat{Q}^\pi_\theta(s,a)$:

$$\mathcal{T}^{\mathrm{TD}}(r,s') = r + \gamma\big(1 - d(s')\big)Q^\pi_{\theta^-}(s', \pi(s')), \tag{2.2}$$

$$\mathcal{L}_\theta = \mathbb{E}_{(s,a,r,s')}\left[\left(\hat{Q}^\pi_\theta(s,a) - \mathcal{T}^{\mathrm{TD}}(r,s')\right)^2\right]. \tag{2.3}$$

In continuous action space, we use a neural network to approximate $\max_a \hat{Q}^\pi_\theta(s,a)$, by learning a parameterized function π_ϕ to minimize the negative Q-value:

$$\mathcal{L}_\phi = -\hat{Q}^\pi_\theta(s, \pi_\phi(s)). \tag{2.4}$$

Model-Based Value Expansion (MVE)

MVE forms TD targets by combining a short term value and a long term value.

- short term: estimate formed by unrolling the model dynamics.
- long term: estimate using the learned $Q^{\pi}_{\theta^-}$ function.

Component	Return
transition function $\hat{T}_\xi(s,a)$	a successor state s'
termination function $\hat{d}_\xi(s)$	the probability that s is a terminal state
reward function $\hat{r}_\psi(s,a,s')$	a scalar reward

The model is trained to minimize:

$$\mathcal{L}_{\xi,\psi} = \mathbb{E}_{(s,a,r,s')}\left[\left\|\hat{T}_\xi(s,a) - s'\right\|_2^2 + D_{KL}\left(d(s')\middle\|\hat{d}_\xi(\hat{T}_\xi(s,a))\right) + \left(\hat{r}_\psi(s,a,s') - r\right)^2\right]. \tag{2.5}$$

Model-Based Value Expansion (MVE)

Improved Q-learning target:

$$s'_0 = s', \quad a'_i = \pi(s'_i), \quad s'_i = \hat{T}_\xi(s'_{i-1}, a'_{i-1}),$$
$$D_i = d(s') \prod_{j=1}^{i} (1 - \hat{d}_\xi(s'_j)), \tag{2.6}$$

$$\mathcal{T}_H^{\mathrm{MVE}}(r, s') = r + \left(\sum_{i=1}^{H} D^i \gamma^i \hat{r}_\psi(s'_{i-1}, a'_{i-1}, s'_i) \right) + D^{H+1} \gamma^{H+1} Q^\pi_{\theta^-}(s'_H, a'_H). \tag{2.7}$$

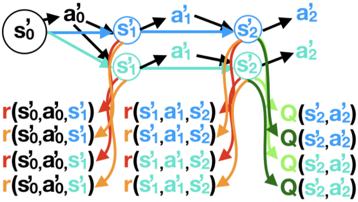
when $H = 0$, MVE reduces to TD-learning (i.e., $\mathcal{T}^{\mathrm{TD}} = \mathcal{T}_0^{\mathrm{MVE}}$).
When the model is perfect and the learned Q-function has similar bias on all states and actions, the MVE target with rollout horizon H will decrease the target error by a factor of γ^{2H}.

Stochastic Ensemble Value Expansion

Maintain an ensemble of parameters for Q-function, reward function, and model: $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_L\}$, $\boldsymbol{\psi} = \{\psi_1, \ldots, \psi_N\}$, and $\boldsymbol{\xi} = \{\xi_1, \ldots, \xi_M\}$, respectively. Each parameterization is initialized independently and trained on different subsets of the data in each minibatch.

Roll out an H step trajectory with each of the M models, $\tau^{\xi_1}, \ldots, \tau^{\xi_M}$. Each trajectory consists of $H+1$ states, $\tau_0^{\xi_m}, \ldots, \tau_H^{\xi_m}$, which correspond to $s'_0, \ldots, s'_H$ in Equation (2.6) with the transition function parameterized by ξ_m. Similarly, Use the N reward functions and L Q-functions to evaluate Equation (2.7) for each τ^{ξ_m} at every rollout-length $0 \leq i \leq H$. This gives us $M \cdot N \cdot L$ different values of $\mathcal{T}_i^{\mathrm{MVE}}$ for each rollout-length i.

Stochastic Ensemble Value Expansion



$$\mathcal{T}_2^{\mathrm{MVE}} = \left\{ \begin{array}{l} r + \gamma r(s'_0,a'_0,s'_1) + \gamma^2 r(s'_1,a'_1,s'_2) + \gamma^3 Q(s'_2,a'_2) \\ r + \gamma r(s'_0,a'_0,s'_1) + \gamma^2 r(s'_1,a'_0,s'_2) + \gamma^3 Q(s'_2,a'_2) \\ r + \gamma r(s'_0,a'_0,s'_1) + \gamma^2 r(s'_1,a'_1,s'_2) + \gamma^3 Q(s'_2,a'_2) \\ r + \gamma r(s'_0,a'_0,s'_1) + \gamma^2 r(s'_1,a'_1,s'_2) + \gamma^3 Q(s'_2,a'_2) \\ r + \gamma r(s'_0,a'_0,s'_1) + \gamma^2 r(s'_1,a'_1,s'_2) + \gamma^3 Q(s'_2,a'_2) \\ r + \gamma r(s'_0,a'_0,s'_1) + \gamma^2 r(s'_1,a'_1,s'_2) + \gamma^3 Q(s'_2,a'_2) \\ r + \gamma r(s'_0,a'_0,s'_1) + \gamma^2 r(s'_1,a'_1,s'_2) + \gamma^3 Q(s'_2,a'_2) \\ r + \gamma r(s'_0,a'_0,s'_1) + \gamma^2 r(s'_1,a'_1,s'_2) + \gamma^3 Q(s'_2,a'_2) \end{array} \right\}$$

Visualization of how the set of possible values for each candidate target is computed, shown for a length-two rollout with $M, N, L = 2$. Colors correspond to ensemble members. Best viewed in color.

Using these values of $\mathcal{T}_i^{\mathrm{MVE}}$, we can compute the empirical mean $\mathcal{T}_i^{\mu}$ and variance $\mathcal{T}_i^{\sigma}$ for each partial rollout of length i. In order to form a single target, we use an inverse variance weighting of the means:

$$\mathcal{T}_H^{\mathrm{MVE}} = \sum_{i=0}^{H} \frac{\tilde{w}_i}{\sum_j \tilde{w}_j} \mathcal{T}_i^{\mu}, \quad \tilde{w}_i^{-1} = \mathcal{T}_i^{\sigma 2}. \tag{2.8}$$

Deviation of STEVE

Mean-squared error:

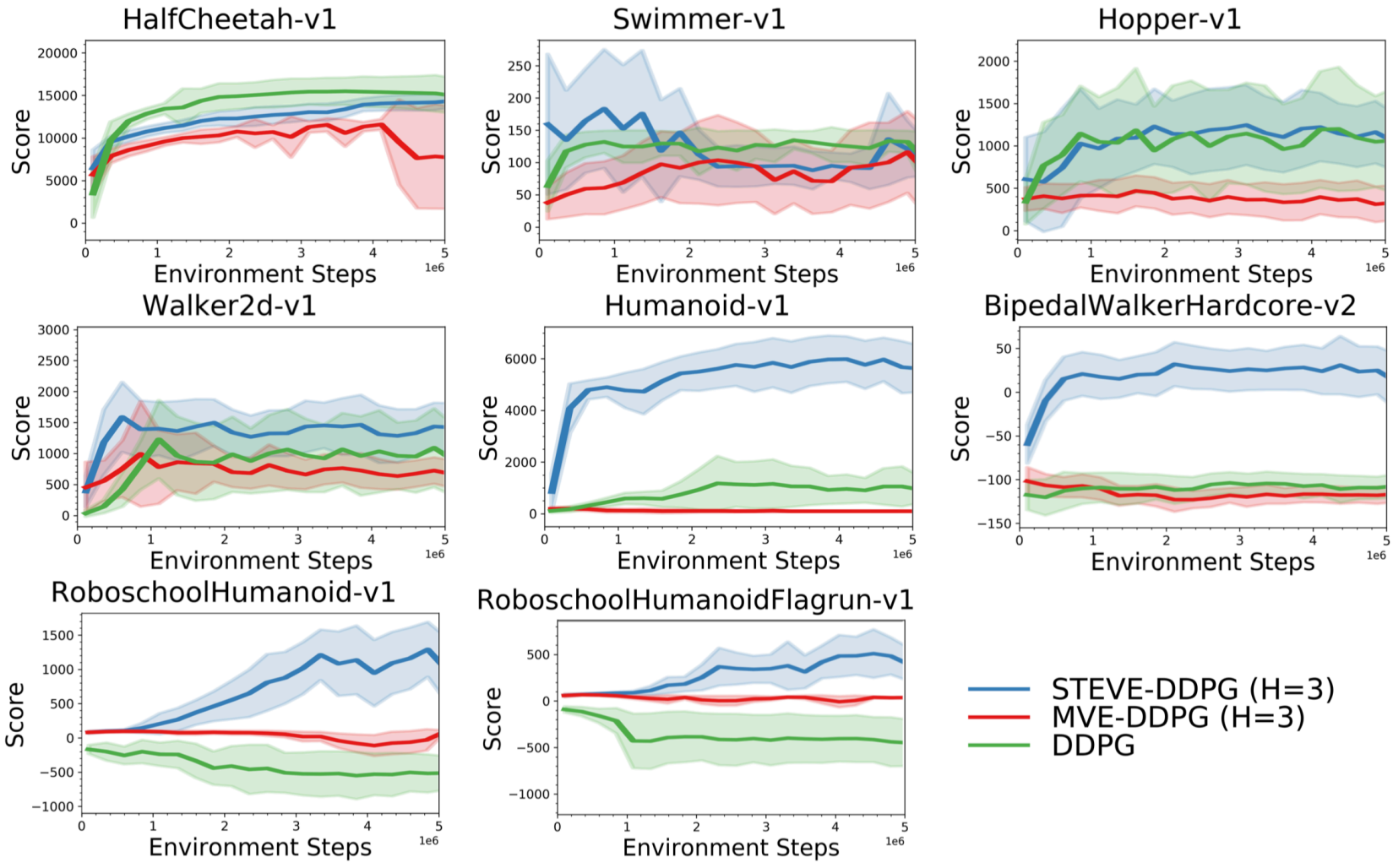
$$\mathbb{E}\left[\left(\sum_{i=0}^{H} w_i \mathcal{T}_i^{\mu} - Q^{\pi}(s,a)\right)^2\right] = \text{Bias}\left(\sum_{i=0}^{H} w_i \mathcal{T}_i^{\mu}\right)^2 + \text{Var}\left(\sum_{i=0}^{H} w_i \mathcal{T}_i^{\mu}\right)^2$$
$$\approx \text{Bias}\left(\sum_{i=0}^{H} w_i \mathcal{T}_i^{\mu}\right)^2 + \sum_{i=0}^{H} w_i^2 \,\text{Var}\left(\mathcal{T}_i^{\mu}\right), \tag{2.9}$$

where $\sum_i w_i = 1$. Our goal is to minimize this with respect to w_i. It is difficult to estimate bias terms, ignore the bias terms and minimize the weighted sum of variances:

$$\sum_{i=0}^{H} w_i^2 \,\text{Var}\left(\mathcal{T}_i^{\mu}\right), \tag{2.10}$$

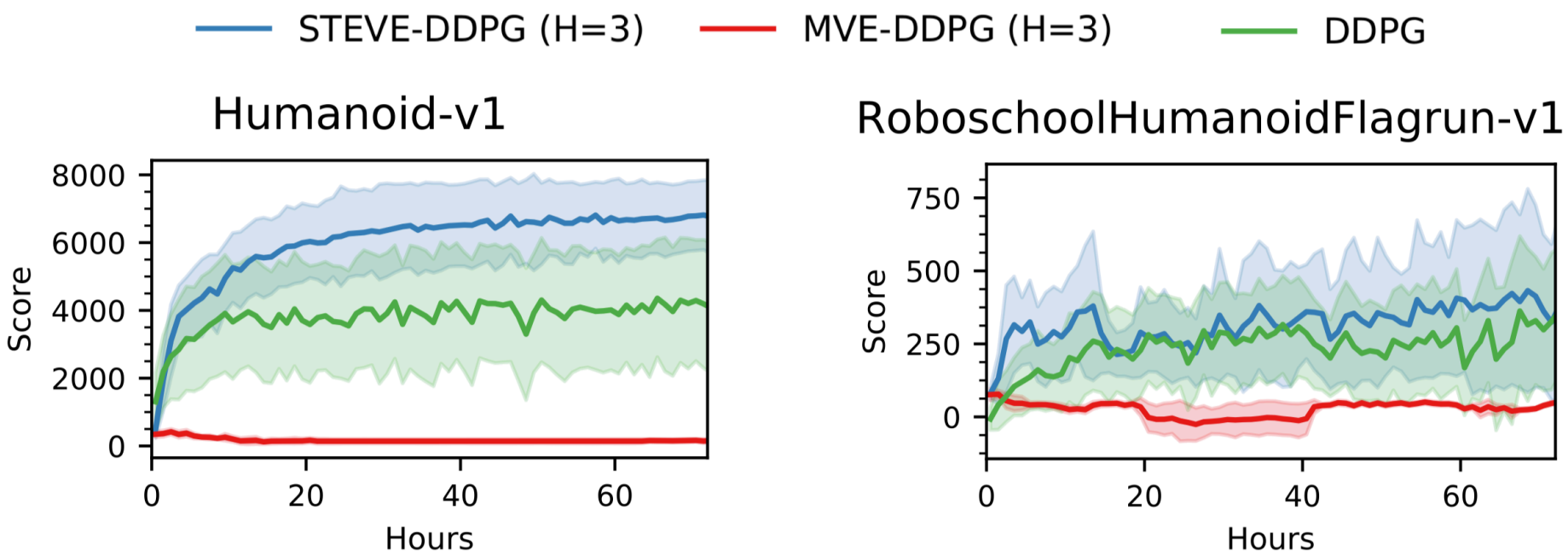
the state-of-the-art results of this approximation are $w_i \propto \frac{1}{\text{Var}\left(\mathcal{T}_i^{\mu}\right)}$.

Experiment Results



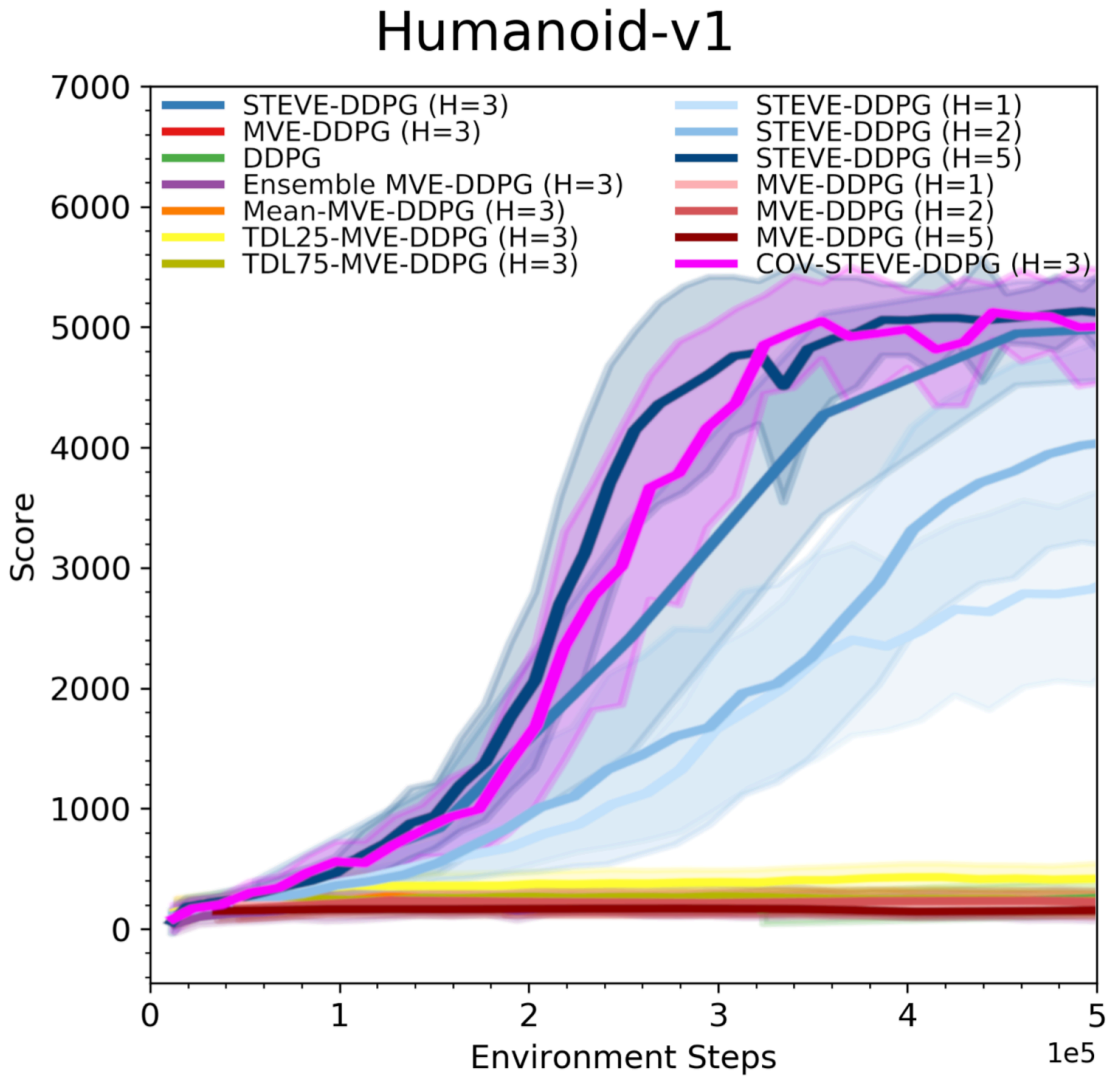
Learning curves comparing sample efficiency of STEVE to both model-free and model-based baselines. Each experiment was run four times.

Experiment Results



Comparison of wall-clock time between STEVE and baselines. Each experiment was run three times.

Ablation Study



Ablation experiments on variation of methods. Each experiment was run twice.

Harnessing Structures for Value-Based Planning and Reinforcement Learning

Yuzhe Yang* Guo Zhang* Zhi Xu* Dina Katabi

Computer Science and Artificial Intelligence Lab

Massachusetts Institute of Technology

ICLR 2020

*Equal contribution

Motivation

- High dimensional space (e.g., images) significantly hamper the efficiency and applicability of the vanilla value iteration.
- The Q-value function is intrinsically induced by the underlying system dynamics (e.g., partial differential equations).
- States and actions may also contain latent features (e.g., similar states could have similar optimal actions).

Definition

Consider about recovering a full data matrix, based on potentially incomplete and noisy observations of its elements.

Given:

- an unknown data matrix $X \in \mathbb{R}^{n \times m}$.
- a set of observed entries Ω. (assume that each entry is observed independently with probability $p \in (0, 1]$).
- the observation could be noisy, where the noise is assumed to be mean zero.

Objective:

$$\min_{\hat{M} \in \mathbb{R}^{n \times m}} \frac{1}{2} \sum_{(i,j) \in \Omega} \left(\hat{M}_{ij} - X_{ij} \right)^2. \quad \text{(Frobenius norm)} \tag{3.1}$$

Matrix estimation with low-rank structure:

$$\min_{\hat{M}\in\mathbb{R}^{n\times m}} \frac{1}{2}\sum_{(i,j)\in\Omega}\left(\hat{M}_{ij}-X_{ij}\right)^2+\lambda\left\|\hat{M}\right\|_*. \tag{3.2}$$

Since the nuclear norm $\|\cdot\|_*$ is a convex relaxation of the rank, the convex optimization approaches favor solutions that are with small reconstruction errors and in the meantime being relatively low-rank.

At the t-th iteration, instead of a full pass over all state-action pairs:

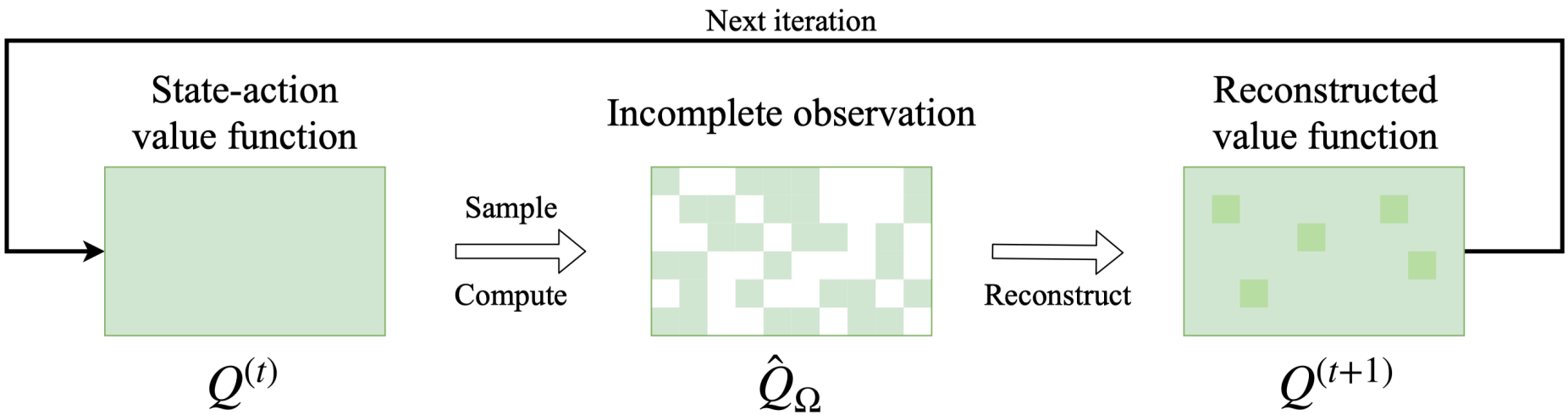
1. SVP first randomly selects a subset Ω of the state-action pairs. In particular, each state-action pair in $\mathcal{S} \times \mathcal{A}$ is observed (i.e., included in Ω) independently with probability p.
2. For each selected (s, a), the intermediate $Q(s, a)$ is computed based on the Q-value iteration:

$$\hat{Q}(s,a) \leftarrow \sum_{s'} P(s'|s,a)\left(r(s,a) + \gamma \max_{a'} Q(s',a')\right), \quad \forall (s,a) \in \Omega. \quad (3.3)$$

3. The current iteration then ends by reconstructing the full Q matrix with matrix estimation, from the set of observations in Ω. That is:

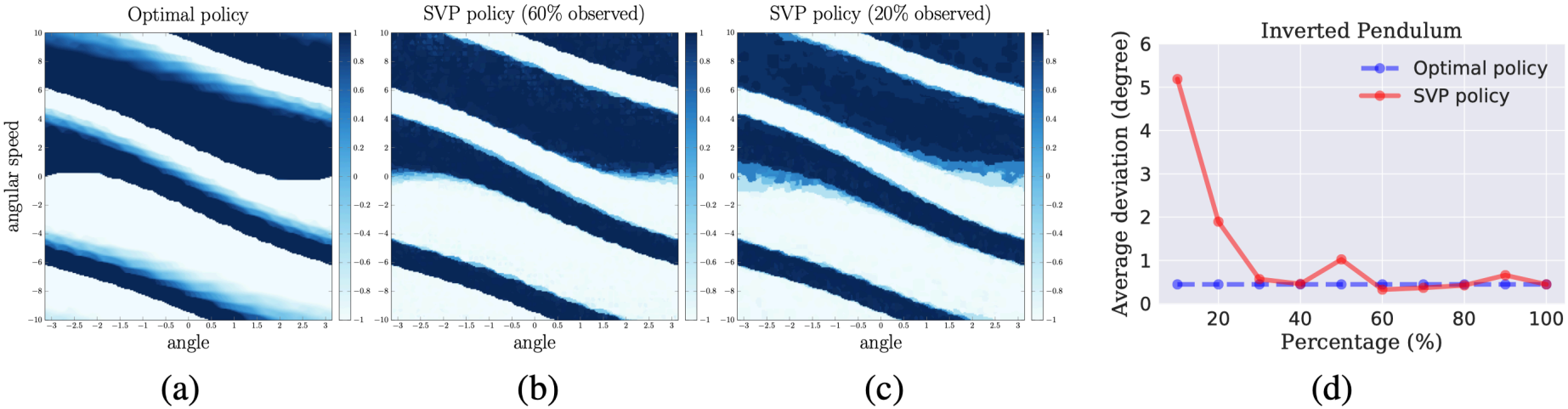
$$Q^{(t+1)} = \mathrm{ME}\left(\left\{\hat{Q}(s,a)\right\}_{(s,a)\in\Omega}\right). \quad (3.4)$$

Structured Value-Based Planning

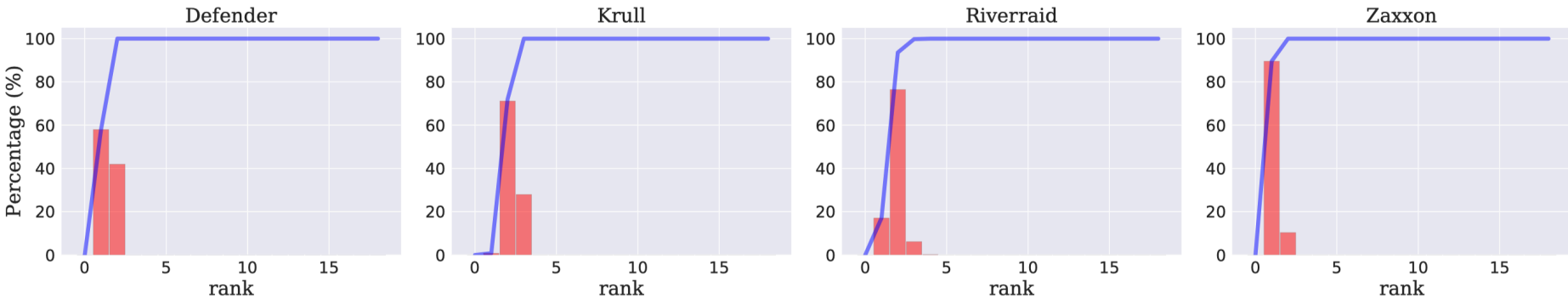


An illustration of the proposed SVP algorithm for leveraging low-rank structures.

Empirical Evaluation

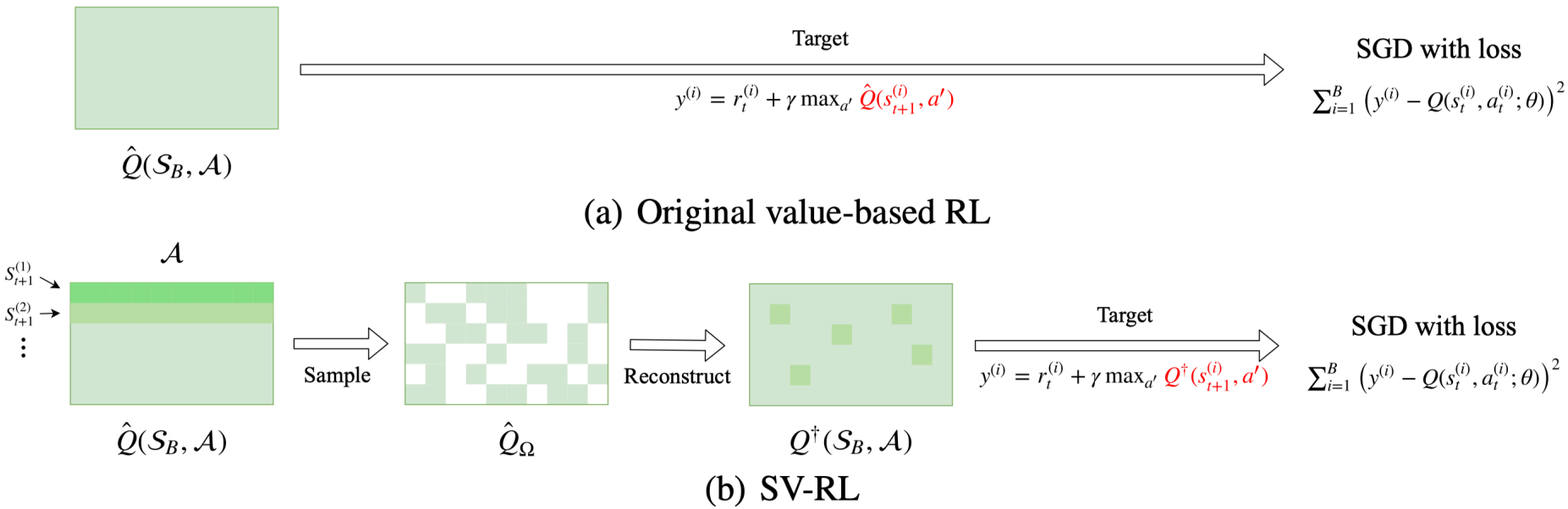


Performance comparison between optimal policy and the proposed SVP policy.



Approximate rank of different Atari games: histogram (red) and empirical CDF (blue).

Structured Value-Based RL



An illustration of the proposed SV-RL scheme, compared to the original value-based RL.

Algorithm 1: Structured Value-based RL (SV-RL)

follow the chosen value-based RL method (e.g., DQN) as usual.

except that for model updates with gradient descent, **do**

`/* exploit structure via matrix estimation*/`

sample a set Ω of state-action pairs from $\mathcal{S}_B \times \mathcal{A}$. In particular, each state-action pair in $\mathcal{S}_B \times \mathcal{A}$ is observed (i.e., included in Ω) with probability p, independently.

evaluate every state-action pair in Ω via $\hat{Q}$, where $\hat{Q}$ is the network that would be used to form the targets $\{y^{(i)}\}_{i=1}^{B}$ in the original value-based methods (cf. Eq. (3)).

based on the evaluated values, reconstruct a matrix $Q^\dagger$ with ME, i.e.,

$$Q^\dagger = \mathrm{ME}\big(\{\hat{Q}(s,a)\}_{(s,a)\in\Omega}\big).$$

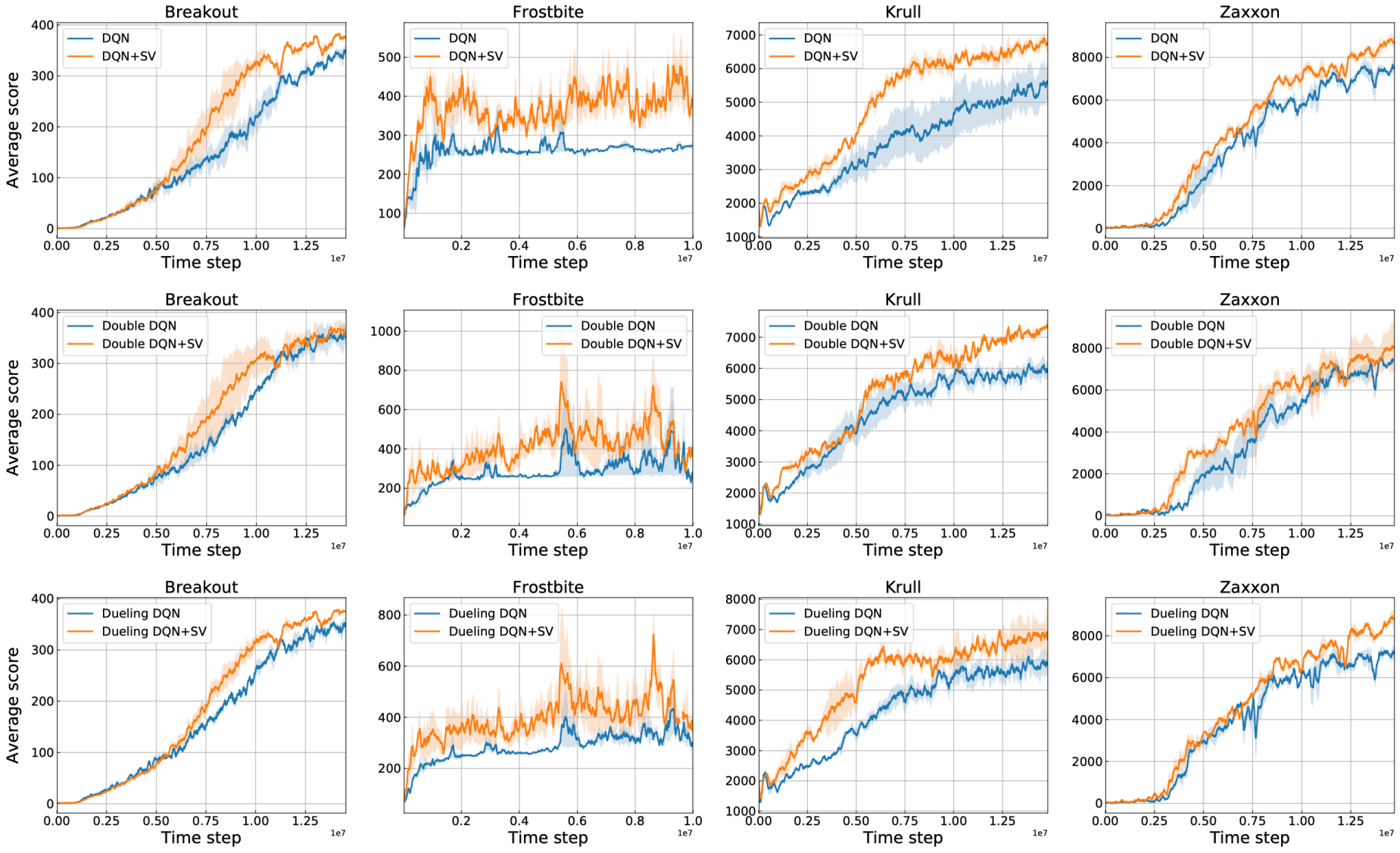
`/* new targets with reconstructed` $Q^\dagger$ `for the gradient step*/`

replace $\hat{Q}$ in Eq. (3) with $Q^\dagger$ to evaluate the targets $\{y^{(i)}\}_{i=1}^{B}$, i.e.,

$$\text{SV-RL Targets:} \quad y^{(i)} = r_t^{(i)} + \gamma \max_{a'} Q^\dagger(s_{t+1}^{(i)}, a'). \tag{4}$$

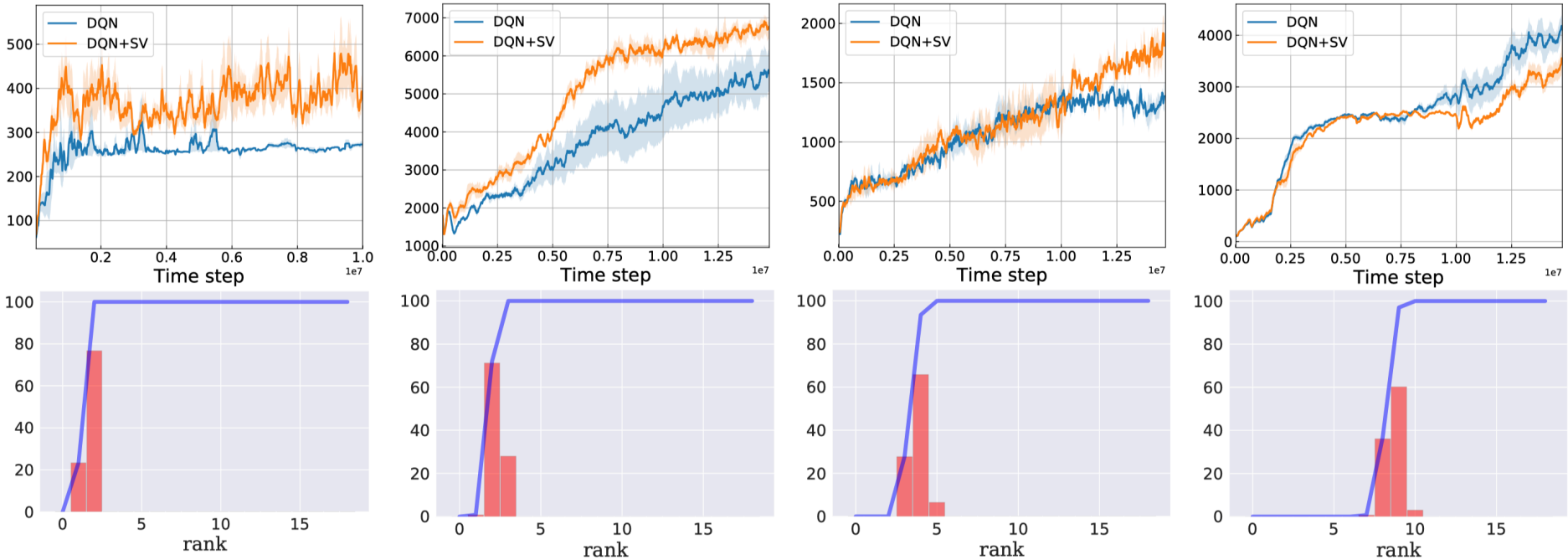
update the Q network with the original targets replaced by the SV-RL targets.

Empirical Evaluation



Results of SV-RL on various value-based deep RL techniques.
First row: results on DQN. **Second row:** results on double DQN. **Third row:** results on dueling DQN.

Diagnose and Interpret Performance



(a) Frostbite (better) (b) Krull (better) (c) Alien (slightly better) (d) Seaquest (worse)

Interpretation of deep RL results. We plot games where the SV-based method performs differently. More structured games (with lower rank) can achieve better performance with SV-RL.

SEED RL: Scalable and Efficient Deep-RL with Accelerated Central Inference

Lasse Espeholt[†] Raphaël Marinier[†] Piotr Stanczyk[†]

Ke Wang Marcin Michalski

Brain Team, Google Research

ICLR 2020

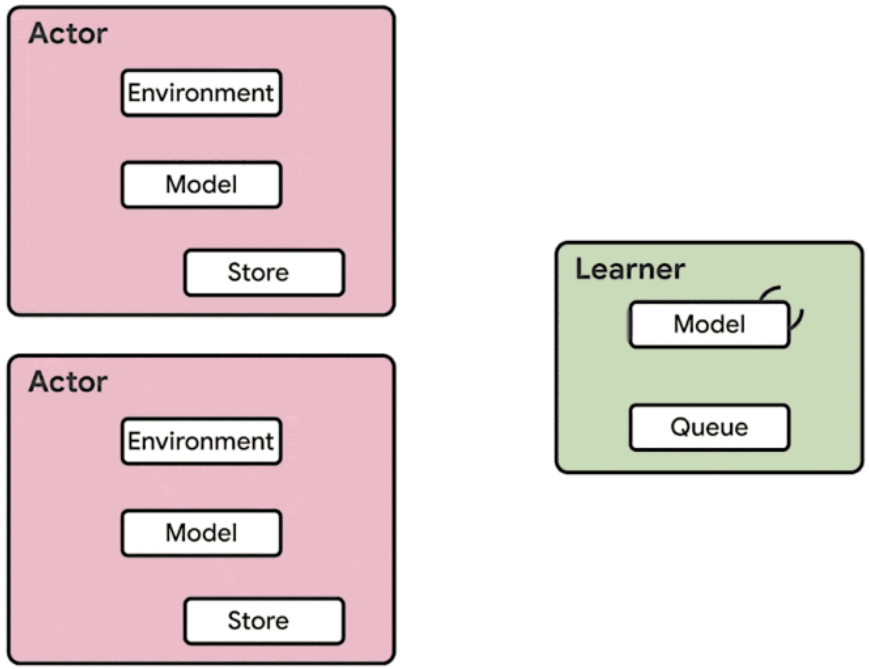
[†] Equal contribution

Motivation

Shortages of IMPALA[3]:

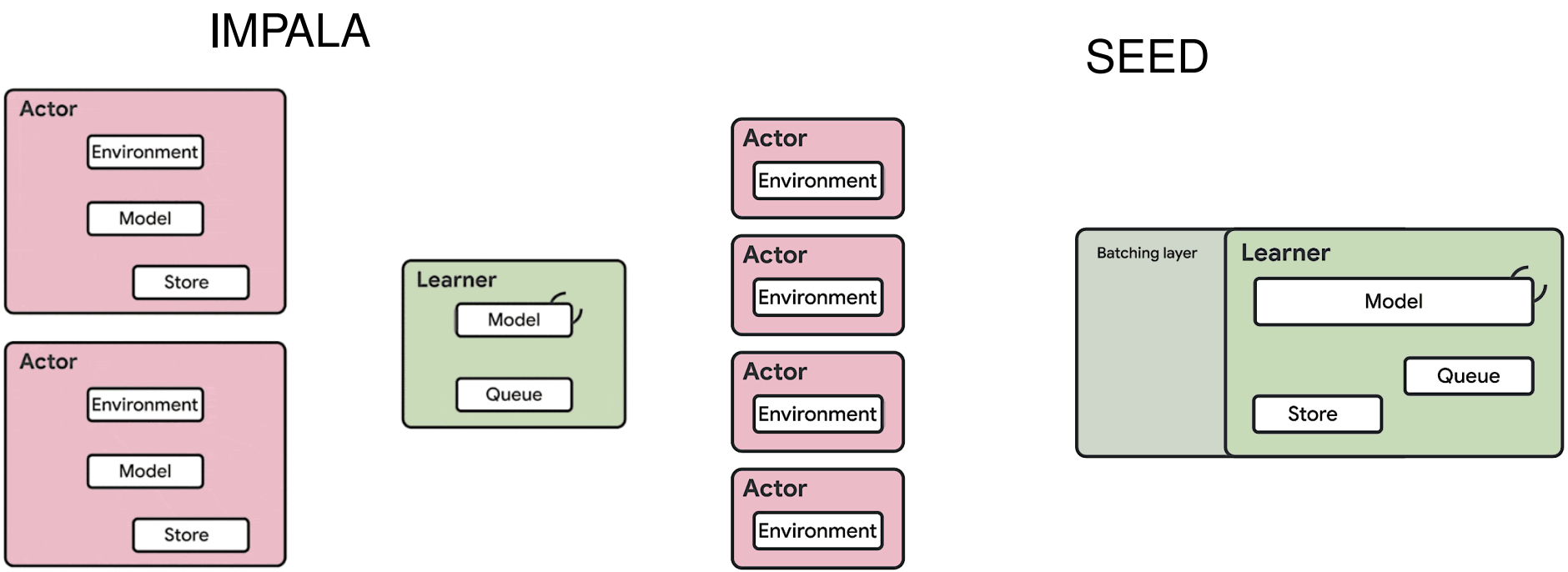
- using CPUs for network inference
- inefficient resource utilization
  - single-thread environment
  - parallel network inference
- bandwidth requirements

Architecture of IMPALA

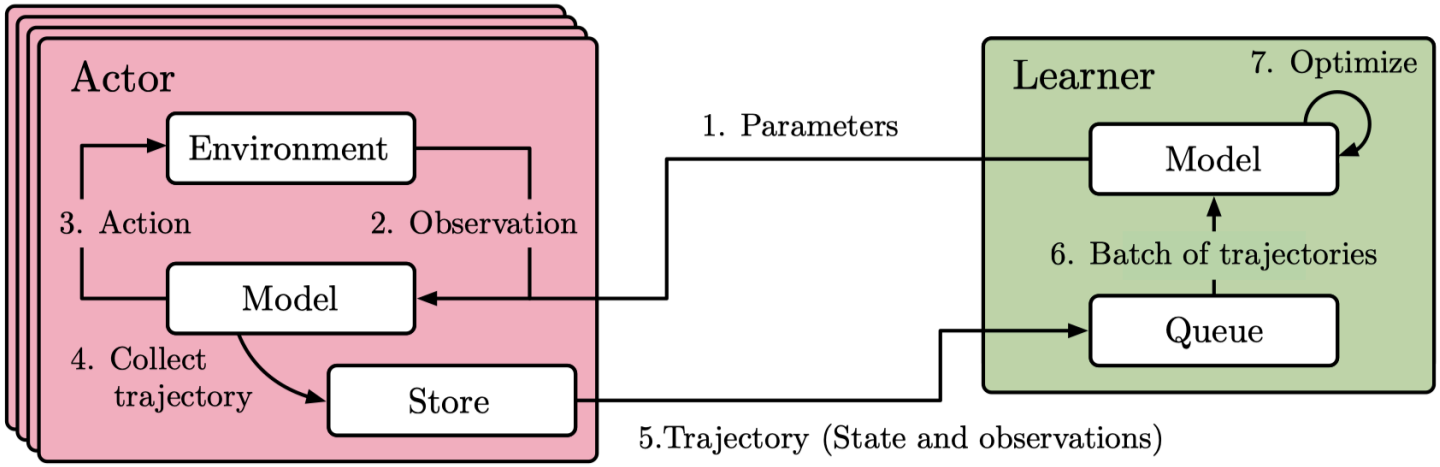


[3] Lasse Espeholt, et al. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. ICML 2018.

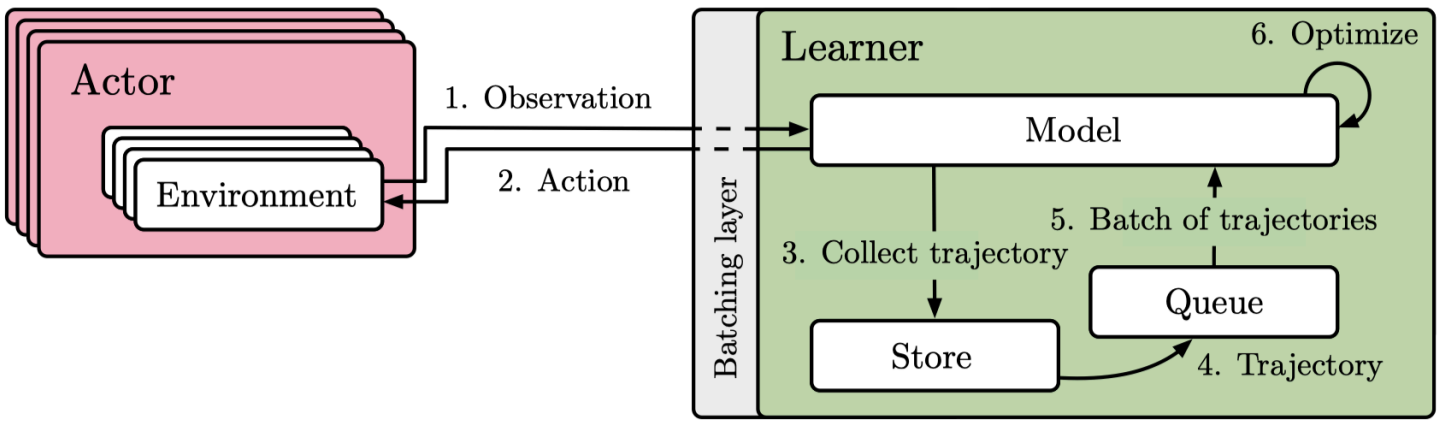
IMPALA v.s. SEED



IMPALA v.s. SEED



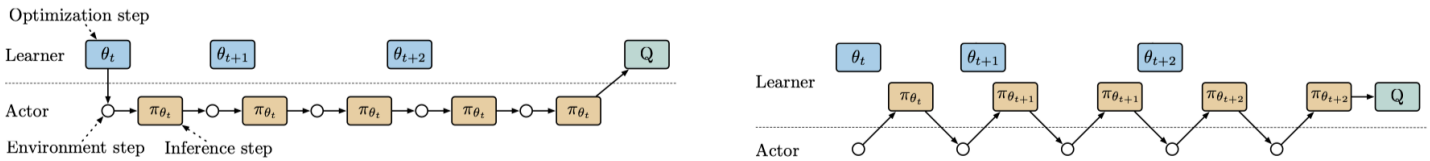
(a) IMPALA architecture (distributed version)



(b) SEED architecture, see detailed replay architecture in Figure 3.

Figure 1: Overview of architectures

IMPALA v.s. SEED



(a) Off-policy in IMPALA. For the entire trajectory the policy stays the same. By the time the trajectory is sent to the queue for optimization, the policy has changed twice.

(b) Off-policy in SEED. Optimizing a model has immediate effect on the policy. Thus, the trajectory consists of actions sampled from many different policies $(\pi_{\theta_t}, \pi_{\theta_{t+1}}, ...)$.

Figure 2: Variants of "near on-policy" when evaluating a policy π while asynchronously optimizing model parameters θ.

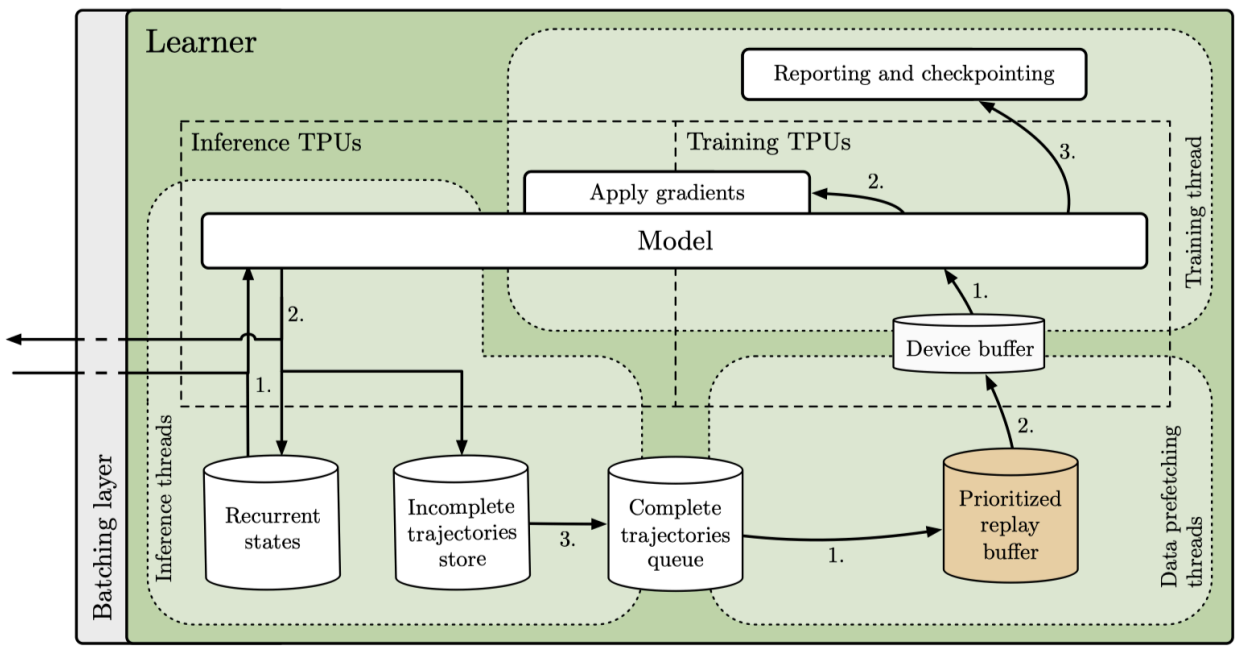


Figure 3: Detailed Learner architecture in SEED (with an optional replay buffer).

Contributions

- Eliminate any neural network related computations from the actors. (increasing the model size)
- Fully utilize both the accelerators on the learner and CPUs on the actors.
- Reduce bandwidth requirements by as much as 99%.
- Use streaming gRPC that has minimal latency and minimal overhead and integrate batching into the server module.

Contributions

- Eliminate any neural network related computations from the actors. (increasing the model size)
- Fully utilize both the accelerators on the learner and CPUs on the actors.
- Reduce bandwidth requirements by as much as 99%.
- Use streaming gRPC that has minimal latency and minimal overhead and integrate batching into the server module.

Stablility and Speed Results

DeepMind Lab and V-trace

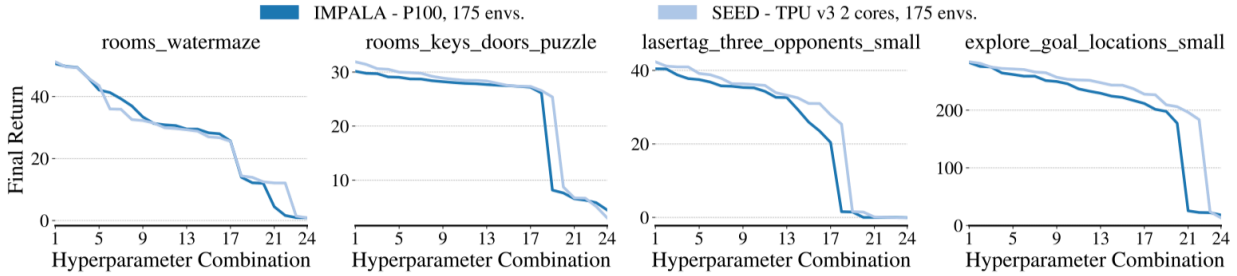


Figure 4: Comparison of IMPALA and SEED under the exact same conditions (175 actors, same hyperparameters, etc.) The plots show hyperparameter combinations sorted by the final performance across different hyperparameter combinations.

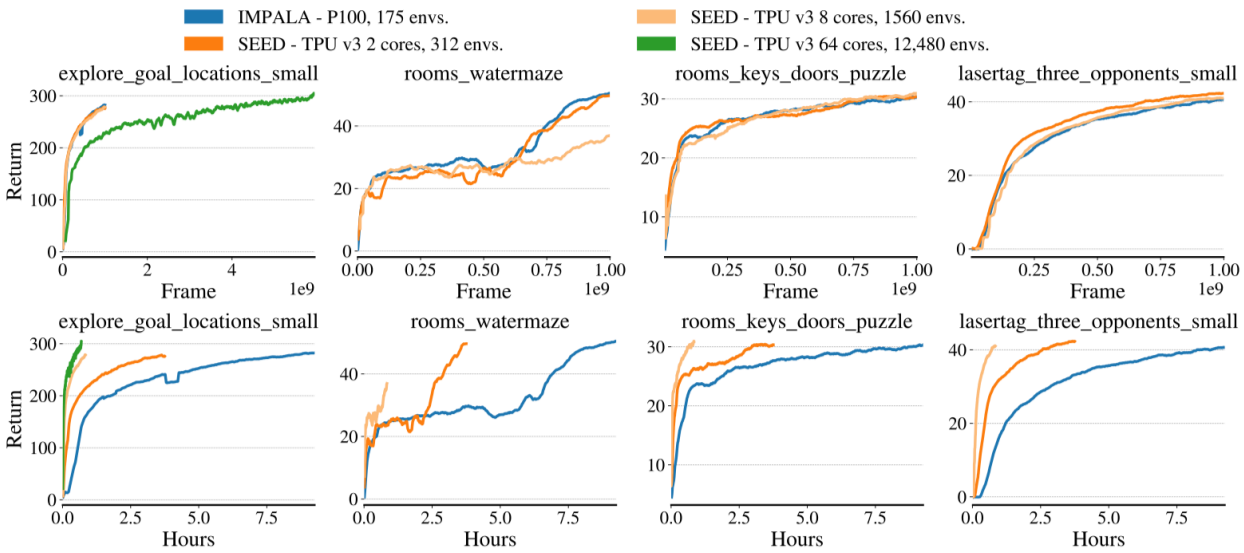


Figure 5: Training on 4 DeepMind Lab tasks. Each curve is the best of the 24 runs based on final return following the evaluation procedure in Espeholt et al. (2018). Sample complexity is maintained up to 8 TPU v3 cores, which leads to 11x faster training than the IMPALA baseline. **Top Row:** X-axis is per frame (number of frames = 4x number of steps). **Bottom Row:** X-axis is hours.

Stablility and Speed Results

Google Research Football and V-trace

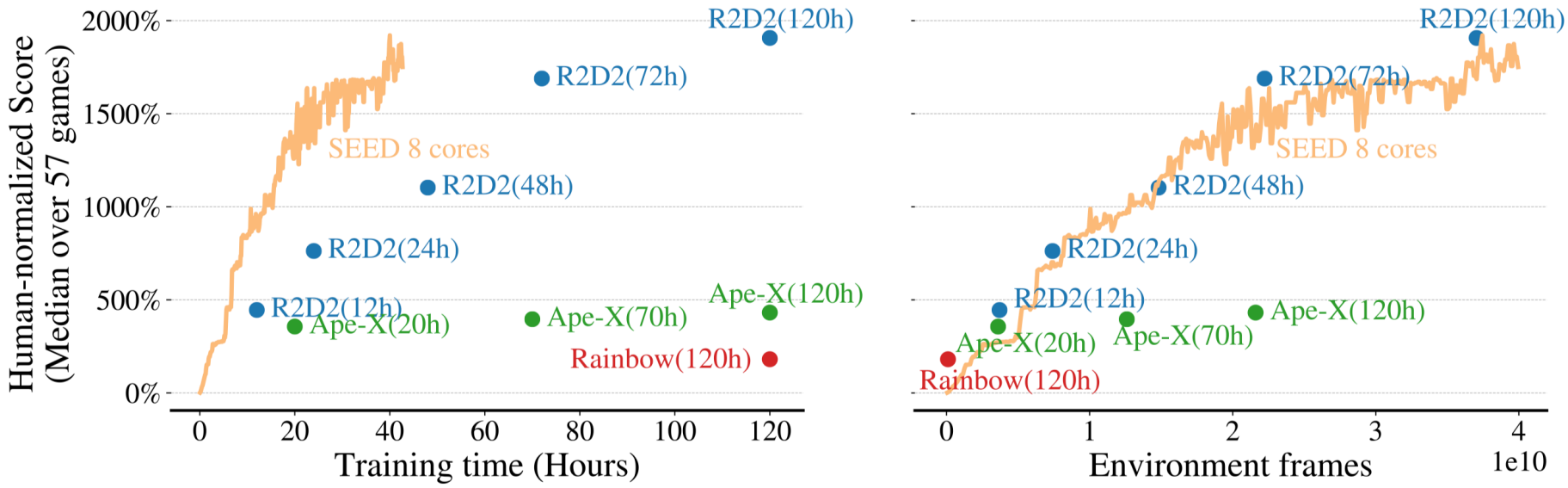


Figure 6: Median human-normalized score on Atari-57 for SEED and related agents. SEED was run with 1 seed for each game. All agents use up to 30 random no-ops for evaluation. **Left:** X-axis is hours **Right:** X-axis is environment frames (a frame is 1/4th of an environment step due to action repeat). SEED reaches state of the art performance **3.1x** faster (wall-time) than R2D2.

Performance Results

Architecture	Accelerators	Environments	Actor CPUs	Batch Size	FPS	Ratio
DeepMind Lab						
IMPALA	Nvidia P100	176	176	32	30K	—
SEED	Nvidia P100	176	44	32	19K	**0.63x**
SEED	TPU v3, 2 cores	312	104	32	74K	**2.5x**
SEED	TPU v3, 8 cores	1560	520	48[1]	330K	**11.0x**
SEED	TPU v3, 64 cores	12,480	4,160	384[1]	2.4M	**80.0x**
Google Research Football						
IMPALA, Default	2 x Nvidia P100	400	400	128	11K	—
SEED, Default	TPU v3, 2 cores	624	416	128	18K	**1.6x**
SEED, Default	TPU v3, 8 cores	2,496	1,664	160[3]	71K	**6.5x**
SEED, Medium	TPU v3, 8 cores	1,550	1,032	160[3]	44K	—
SEED, Large	TPU v3, 8 cores	1,260	840	160[3]	29K	—
SEED, Large	TPU v3, 32 cores	5,040	3,360	640[3]	114K	**3.9x**
Arcade Learning Environment						
R2D2	Nvidia V100	256	N/A	64	85K[2]	—
SEED	Nvidia V100	256	55	64	67K	**0.79x**
SEED	TPU v3, 8 cores	610	213	64	260K	**3.1x**
SEED	TPU v3, 8 cores	1200	419	256	440K[4]	**5.2x**

[1] 6/8 cores used for training. [2] Each of the 256 R2D2 actors run at 335 FPS (information from the R2D2 authors). [3] 5/8 cores used for training. [4] No frame stacking.

Table 1: Performance of SEED, IMPALA and R2D2.

Performance Results

Architecture	Accelerators	SMM	Median	Max
Scoring reward				
IMPALA	2 x Nvidia P100	Default	-0.74	0.06
SEED	TPU v3, 2 cores	Default	-0.72	-0.12
SEED	TPU v3, 8 cores	Default	-0.83	-0.02
SEED	TPU v3, 8 cores	Medium	-0.74	0.12
SEED	TPU v3, 8 cores	Large	**-0.69**	**0.46**
SEED	TPU v3, 32 cores	Large	n/a	**4.76**[1]
Checkpoint reward				
IMPALA	2 x Nvidia P100	Default	**-0.27**	1.63
SEED	TPU v3, 2 cores	Default	-0.44	1.64
SEED	TPU v3, 8 cores	Default	-0.68	1.55
SEED	TPU v3, 8 cores	Medium	-0.52	1.76
SEED	TPU v3, 8 cores	Large	-0.38	**1.86**
SEED	TPU v3, 32 cores	Large	n/a	**7.66**[1]

[1] 32 core experiments trained on 4B frames with a limited sweep.

Table 2: Google Research Football "Hard" using two kinds of reward functions. For each reward function, 40 hyperparameter sets ran with 3 seeds each which were averaged after 500M frames of training. The table shows the median and maximum of the 40 averaged values. This is a similar setup to Kurach et al. (2019) although we ran 40 hyperparameter sets vs. 100 but did not rerun our best models using 5 seeds.

Cost Comparisons

Resource	Cost per hour
CPU core	$0.0475
Nvidia Tesla P100	$1.46
TPU v3 core	$1.00

Table 3: Cost of cloud resources as of Sep. 2019.

Model	Actors	CPUs	Envs.	Speed	Cost/1B	Cost ratio
IMPALA						
Default	176	176	176	30k	$90	—
Medium	130	130	130	16.5k	$128	—
Large	100	100	100	7.3k[1]	$236	—
SEED						
Default	26	104	312	74k	$25	**3.60**
Medium	12	48	156	34k	$35	**3.66**
Large	6	24	84	16k	$54	**4.37**

[1] The batch size was lowered from 32 to 16 due to limited memory on Nvidia P100.

Table 4: Training cost on DeepMind Lab for 1 billion frames.

Model	Actors	CPUs	Envs.	Speed	Cost/1B	Cost ratio
IMPALA						
Default	400	400	400	11k	$553	—
Medium	300	300	300	7k	$681	—
Large	300	300	300	5.3k	$899	—
SEED						
Default	52	416	624	17.5k	$345	**1.68**
Medium	31	248	310	10.5k	$365	**1.87**
Large	21	168	252	7.5k	$369	**2.70**

Table 5: Training cost on Google Research Football for 1 billion frames.

Generative Adversarial Imitation Learning

Jonathan Ho Stefano Ermon

Stanford University

NeurIPS 2016

Motivation

Two main approaches of imitation learning:

- Behavioral cloning: learn a policy as a supervised learning problem over state-action pairs from expert trajectories.

  shortage: need large amounts of data due to compounding error caused by covariate shift.

- Inverse reinforcement learning (IRL): find a cost function under which the expert is uniquely optimal.

  shortage: extremely expensive to run.

extended real numbers $\bar{\mathbb{R}} = \mathbb{R} \bigcup \{\infty\}$

finite state space $\mathcal{S}$

finite action space $\mathcal{A}$

set of all stationary stochastic policies Π

dynamics model $P(s'|s,a)$

expectation:

$$\mathbb{E}_\pi[c(s,a)] \triangleq \mathbb{E}_{a_t \sim \pi(\cdot|s_t)} \left[\sum_{k=0}^{\infty} \gamma^t c(s_t, a_t) \right]. \tag{5.1}$$

expert policy π_E

Maximum causal entropy IRL:

$$\underset{c\in\mathcal{C}}{\text{maximize}} \left[\min_{\pi\in\Pi} \left(-H(\pi) + \mathbb{E}_{\pi}[c(s,a)]\right) - \mathbb{E}_{\pi_E}[c(s,a)]\right], \tag{5.2}$$

where $\mathcal{C}$ is the family of cost functions, $H(\pi) = \mathbb{E}_{\pi}[-\log \pi(a|s)]$ is the γ-discounted causal entropy of the policy π.

Reinforcement learning procedure:

$$\text{RL}(c) = \arg\min_{\pi\in\Pi} \left(-H(\pi) + \mathbb{E}_{\pi}[c(s,a)]\right). \tag{5.3}$$

Characterizing the Induced Optimal Policy

Define an IRL primitive procedure, which finds a cost function such that the expert performs better than all other policies, with the cost regularized by ψ:

$$\mathrm{IRL}_{\psi}(\pi_E) = \underset{c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}}{\arg\max} \left[-\psi(c) + \min_{\pi \in \Pi} \left(-H(\pi) + \mathbb{E}_{\pi}[c(s,a)]\right) - \mathbb{E}_{\pi_E}[c(s,a)] \right]. \tag{5.4}$$

where $\psi : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \bar{\mathbb{R}}$ is a convex cost function regularizer.

We are interested in a policy given by $\mathrm{RL}(\tilde{c})$, where $\tilde{c} \in \mathrm{IRL}_{\psi}(\pi_E)$.

Characterizing the Induced Optimal Policy

北京大学 PEKING UNIVERSITY

Definition 1

Occupancy measure $\rho_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

$$\rho_\pi(s,a) = \pi(a|s) \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\pi). \tag{5.5}$$

We can get $\mathbb{E}_\pi[c(s,a)] = \sum_{s,a} \rho_\pi(s,a) c(s,a)$.

Set of valid occupancy measures $\mathcal{D} = \{\rho_\pi : \pi \in \Pi\}$:

$$\mathcal{D} = \left\{ \rho : \rho \geq 0 \text{ and } \sum_a \rho(s,a) = p_0(s) + \gamma \sum_{s',a} P\big(s\big|s',a\big)\rho(s',a), \forall s \in \mathcal{S} \right\}. \tag{5.6}$$

Characterizing the Induced Optimal Policy

北京大学 PEKING UNIVERSITY

Proposition 1

If $\rho \in \mathcal{D}$, then ρ is the occupancy measure for $\pi_\rho(a|s) = \frac{\rho(s,a)}{\sum_{a'} \rho(s,a')}$, and π_ρ is the only policy whose occupancy measure is ρ.

Proposition 2

$$\mathrm{RL} \circ \mathrm{IRL}_\psi(\pi_E) = \arg\min_{\pi \in \Pi} \left(-H(\pi) + \psi^*(\rho_\pi - \rho_{\pi_E})\right) \tag{5.7}$$

for a function $f : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \bar{\mathbb{R}}$, its convex conjugate $f^* : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \bar{\mathbb{R}}$ is given by $f^*(x) = \sup_{y \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}} x^T y - f(y)$.

Corollary 2.1

If ψ is a constant function, $\tilde{c} \in \mathrm{IRL}_\psi(\pi_E)$, and $\tilde{\pi} \in \mathrm{RL}(\tilde{c})$, then $\rho_{\tilde{\pi}} = \rho_{\pi_E}$.

Characterizing the Induced Optimal Policy

北京大学 PEKING UNIVERSITY

Lemma 1

Let $\bar{H}(\rho) = -\sum_{s,a} \rho(s,a) \log(\rho(s,a)/\sum_{s'} \rho(s,a'))$. Then, $\bar{H}(\rho)$ is strictly concave, and for all $\pi \in \Pi$ and $\rho \in \mathcal{D}$, we have $H(\pi) = \bar{H}(\rho_\pi)$ and $\bar{H}(\rho) = H(\pi_\rho)$.

Lemma 2

If $L(\pi, c) = -H(\pi) + \mathbb{E}_\pi[c(s,a)]$ and $\bar{L}(\rho, c) = -\bar{H}(\rho) + \sum_{s,a} \rho(s,a)c(s,a)$, then, for all cost functions c, $L(pi, c) = \bar{L}(\rho_\pi, c)$ for all policies $\pi \in \Pi$, and $\bar{L}(\rho_\pi, c) = L(\pi, c)$ for all occupancy measures $\rho \in \mathcal{D}$.

Proof of Corollary 2.1

Define $\bar{L}(\rho, c) = -\bar{H}(\rho) + \sum_{s,a} c(s,a)(\rho(s,a) - \rho_E(s,a))$. Given that ψ is a constant function, we have the following, due to Lemma 1:

$$
\begin{aligned}
&\tilde{c} \in \mathrm{IRL}_{\psi}(\pi_E) \\
&= \arg\max_{c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}} \left[\min_{\pi \in \Pi} \left(-H(\pi) + \mathbb{E}_{\pi}[c(s,a)]\right) - \mathbb{E}_{\pi_E}[c(s,a)] + const \right] \\
&= \arg\max_{c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}} \left[\min_{\rho \in \mathcal{D}} \left(-\bar{H}(\rho) + \sum_{s,a} c(s,a)\rho(s,a) \right) - \sum_{s,a} c(s,a)\rho_E(s,a) \right] \qquad (5.8) \\
&= \arg\max_{c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}} \left[\min_{\rho \in \mathcal{D}} \bar{L}(\rho, c) \right].
\end{aligned}
$$

This is the dual of the optimization problem:

$$
\underset{\rho \in \mathcal{D}}{\text{minimize}} \; -\bar{H}(\rho), \quad \text{subject to } \rho(s,a) = \rho_E(s,a), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}. \qquad (5.9)
$$

Proof of Corollary 2.1

With Lagrangian $\bar{L}$, for which the costs $c(s,a)$ serve as dual variables for equality constraints. Thus, $\tilde{c}$ is a dual optimum for (5.9). Because $\mathcal{D}$ is a convex set and $-\bar{H}$ is convex, strong duality holds; moreover, Lemma 1 guarantees that $-\bar{H}$ is in fact strictly convex, so the primal optimum can be uniquely recovered from the dual optimum via $\tilde{\rho} = \arg\min_{\rho\in\mathcal{D}} \bar{L}(\rho,\tilde{c}) = \arg\min_{\rho\in\mathcal{D}} \left[-\bar{H}(\rho) + \sum_{s,a} \tilde{c}(s,a)\rho(s,a)\right] = \rho_E$, where the first equality indicates that $\tilde{\rho}$ is the unique minimizer of $\bar{L}(\cdot,\tilde{c})$, and the third follows from the constraints in the primal problem (5.9). But if $\tilde{\pi} \in \mathrm{RL}(\tilde{c})$, then, by Lemma 2, its occupancy measure satises $\rho_{\tilde{\pi}} = \tilde{\rho} = \rho_E$.

Characterizing the Induced Optimal Policy

We can deduce the following:

- **IRL is a dual of an occupancy measure matching problem**, and the recovered cost function is the dual optimum.
- **The induced optimal policy is the primal optimum.**

Practical Occupancy Measure Matching

Relax Eq.(5.9) into the following form, motivated by Proposition 2:

$$\underset{\pi}{\text{minimize}}\, d_\psi(\rho_\pi, \rho_E) - H(\pi), \tag{5.10}$$

by modifying the IRL regularizer ψ so that $d_\psi(\rho_\pi, \rho_E) \triangleq \psi^*(\rho_\pi - \rho_{\pi_E})$ smoothly penalizes violations in difference between the occupancy measures.

Practical Occupancy Measure Matching

With indicator function $\delta_{\mathcal{C}} : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \bar{\mathbb{R}}$, defined by $\delta_{\mathcal{C}}(c) = 0$ if $c \in \mathcal{C}$ and $+\infty$ otherwise. Write apprenticeship learning objective as:

$$\begin{aligned}
&\max_{c\in\mathcal{C}} \mathbb{E}_{\pi}[c(s,a)] - \mathbb{E}_{\pi_E}[c(s,a)] \\
=&\max_{c\in\mathbb{R}^{\mathcal{S}\times\mathcal{A}}} -\delta_{\mathcal{C}}(c) + \sum_{s,a} c(s,a)(\rho(s,a) - \rho_E(s,a)) \qquad (5.11)\\
=&\delta^*_{\mathcal{C}}(\rho_\pi - \rho_{\pi_E}).
\end{aligned}$$

Entropy-regularized apprenticeship learning objective:

$$\underset{\pi}{\text{minimize}} \ -H(\pi) + \max_{c\in\mathcal{C}} \mathbb{E}_{\pi}[c(s,a)] - \mathbb{E}_{\pi_E}[c(s,a)]. \qquad (5.12)$$

is equivalent to performing RL following IRL with cost regularizer $\psi = \delta_{\mathcal{C}}$, which forces the implicit IRL procedure to recover a cost function lying in $\mathcal{C}$.

Cost regularizer:

$$\psi_{GA}(c) \triangleq \begin{cases} \mathbb{E}_\pi[g(c(s,a))], & \text{if } c < 0, \\ +\infty, & \text{otherwise.} \end{cases}, \quad \text{where } g(x) = \begin{cases} -x - \log(1 - e^x), & \text{if } x < 0, \\ +\infty, & \text{otherwise.} \end{cases} \tag{5.13}$$

The conjugate function:

$$\psi^*_{GA}(\rho_\pi - \rho_{\pi_E}) = \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_\pi[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))], \tag{5.14}$$

where the maximum ranges over discriminative classifiers $D : \mathcal{S} \times \mathcal{A} \to (0,1)$.

Imitation learning objective:

$$\underset{\pi}{\text{minimize}} \ \psi^*_{GA}(\rho_\pi - \rho_{\pi_E}) - \lambda H(\pi) = D_{JS}(\rho_\pi, \rho_{\pi_E}) - \lambda H(\pi). \tag{5.15}$$

Generative Adversarial Imitation Learning

北京大学 PEKING UNIVERSITY

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

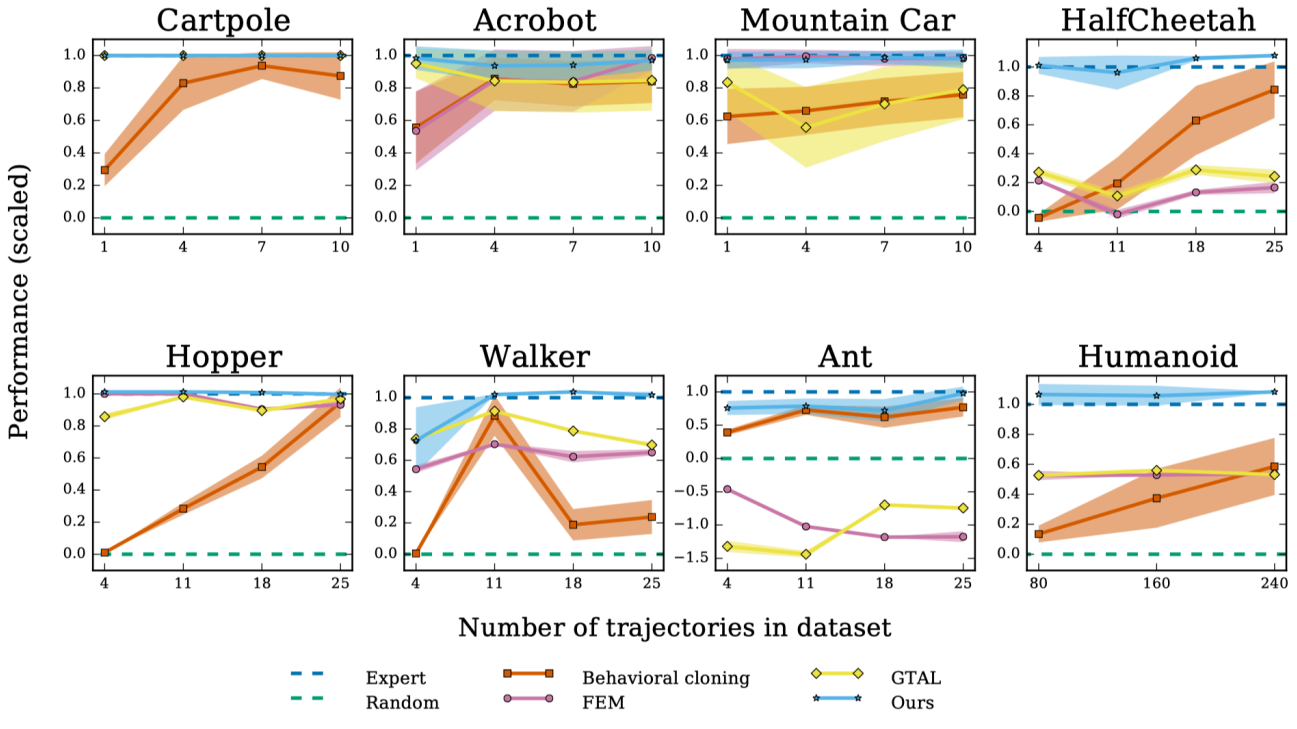
$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \tag{17}$$

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

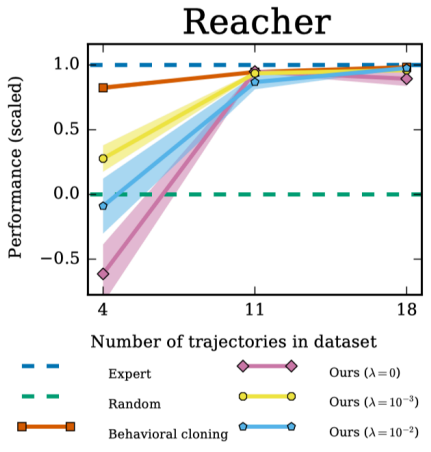
$$\begin{aligned} &\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right] - \lambda \nabla_\theta H(\pi_\theta), \\ &\text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \,|\, s_0 = \bar{s}, a_0 = \bar{a}] \end{aligned} \tag{18}$$

6: **end for**

Experiment Results



(a)



(b)

Figure 1: (a) Performance of learned policies. The y-axis is negative cost, scaled so that the expert achieves 1 and a random policy achieves 0. (b) Causal entropy regularization λ on Reacher.

Causal Confusion in Imitation Learning

Pim de Haan[1] Dinesh Jayaraman[23] Sergey Levine[2]

[1]Qualcomm AI Research, University of Amsterdam

[2]Berkeley AI Research [3]Facebook AI Research

NeurIPS 2018

Motivation

- "Causal misidentification" phenomenon, i.e., access to more information can yield worse performance.
- In supervised learning, it assumes that training and testing distributions are identical, since nuisance correlates continue to hold in the test set.
- In imitation learning, current actions cause future observations often introduces complex new nuisance correlates.

Causal Misidentification

Causal misidentification occurs commonly in natural imitation learning settings, especially when the imitator's inputs include history information.

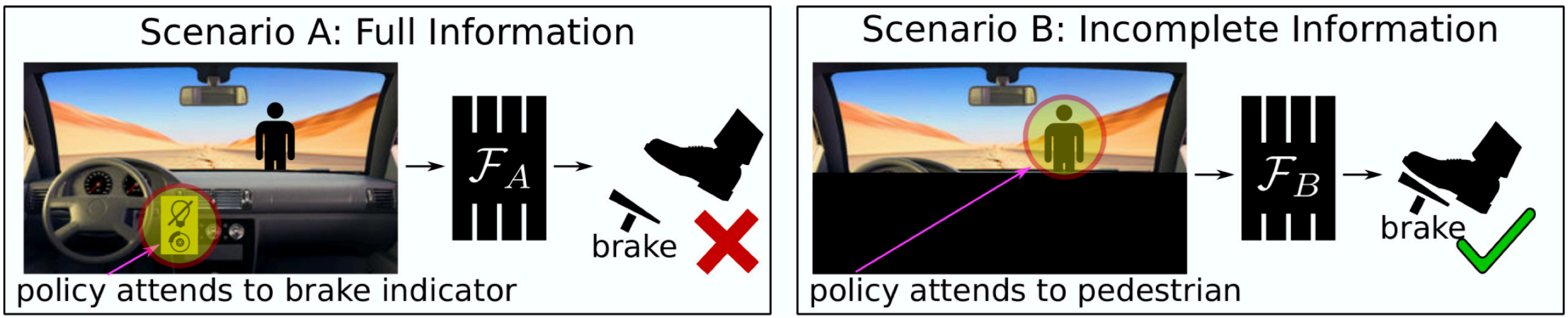


Figure 1: Causal misidentification: *more* information yields worse imitation learning performance. Model A relies on the braking indicator to decide whether to brake. Model B instead correctly attends to the pedestrian.

Even though the brake light is the effect of braking, model A could achieve low training error by misidentifying it as the cause instead.

Causal Dynamics of Imitation

- Expert actions A^t are influenced by **some** information in state X^t, and unaffected by the rest.
- A confounder $Z^t = [X^{t-1}, A^{t-1}]$ influences each state variable in X^t, so that some nuisance variables may still be correlated with A^t among (X^t, A^t) pairs from demonstrations.

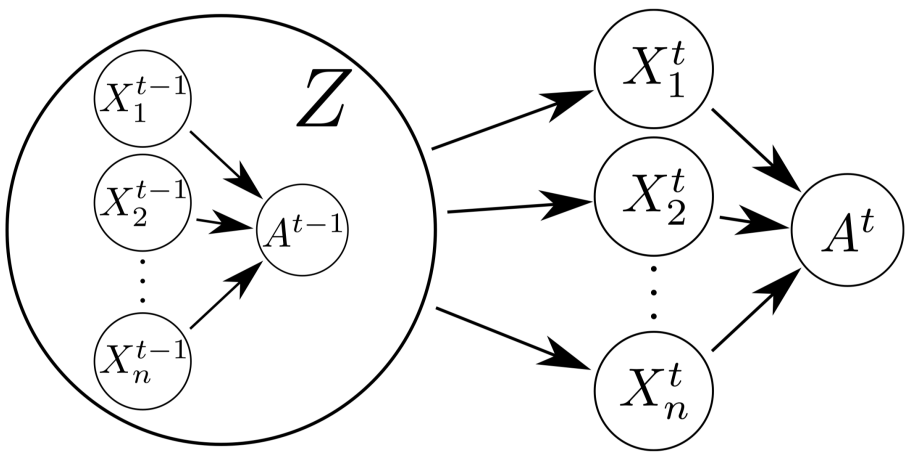


Figure 2: Causal dynamics of imitation. Parents of a node represent its causes.

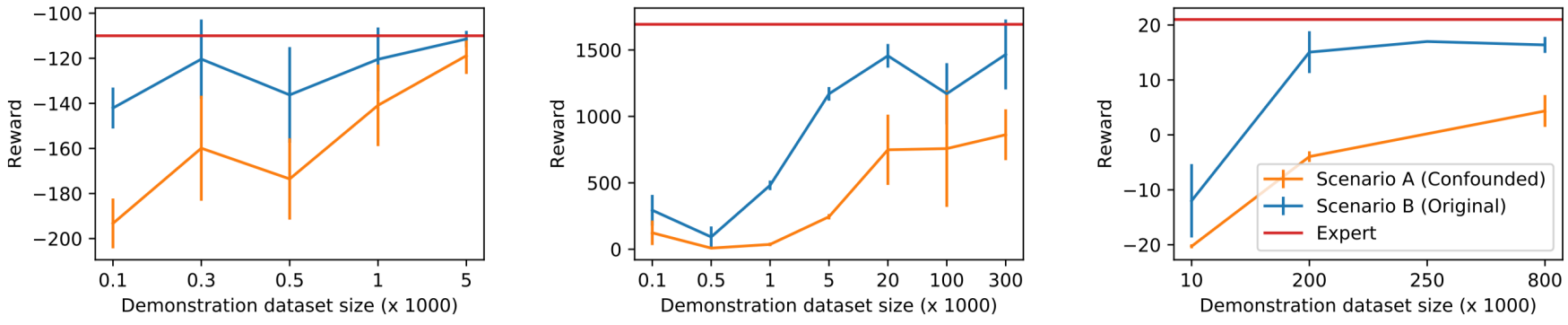
Settings

- In scenario A (called "CONFOUNDED"), the policy sees the augmented observation vector, including the previous action.
- In scenario B (called "ORIGINAL"), the previous action variable is replaced with random noise for low-dimensional observations.

(a) Pong (b) Enduro (c) UpNDown

Figure 3: The Atari environments with indicator of past action (white number in lower left).

Causal Misidentification



(a) MountainCar (b) Hopper (c) Pong

Figure 4: Diagnosing causal misidentification: net reward (y-axis) vs number of training samples (x-axis) for ORIGINAL and CONFOUNDED, compared to expert reward (mean and stdev over 5 runs). Also see Appendix E.

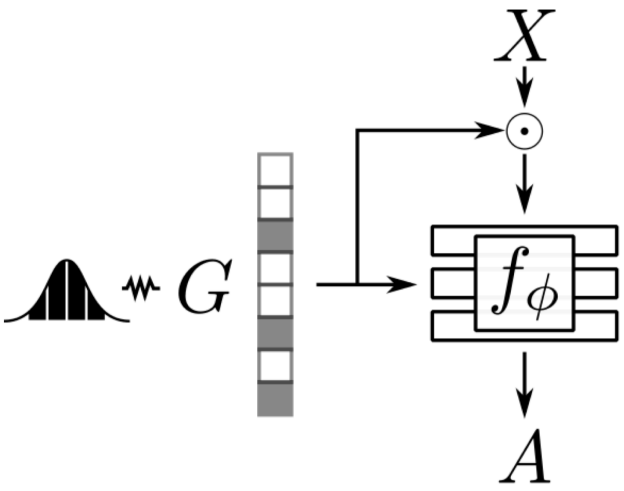


Figure 5: Graph-parameterized policy.

We parameterize the structure G of the causal graph as a vector of n binary variables. We then train a single graph-parameterized policy $\pi_G(X) = f_\phi([X \odot G, G])$, where $\odot$ is element-wise multiplication, and $[\cdot, \cdot]$ denotes concatenation. ϕ are neural network parameters, trained through gradient descent to minimize:

$$\mathbb{E}_G[(f_\phi([X \odot G, G]), A_i)], \tag{6.1}$$

where G is drawn uniformly at random over all 2^n graphs and is a mean squared error loss for the continuous action environments and a cross-entropy loss for the discrete action environments.

Targeted Intervention

Proposed targeted intervention to compute the likelihood $\mathcal{L}(G)$ of each causal graph structure hypothesis G.

Expert query mode

Algorithm 1 Expert query intervention

Input: policy network f_ϕ s.t. $\pi_G(X) = f_\phi([X \odot G, G])$
Initialize $w = 0, \mathcal{D} = \emptyset$.
Collect states $\mathcal{S}$ by executing π_{mix}, the mixture of policies π_G for uniform samples G.
For each X in S, compute disagreement score:
$D(X) = \mathbb{E}_G[D_{KL}(\pi_G(X), \pi_{mix}(X))]$
Select $\mathcal{S}' \subset \mathcal{S}$ with maximal $D(X)$.
Collect state-action pairs $\mathcal{T}$ by querying expert on $\mathcal{S}'$.
for $i = 1 \ldots N$ **do**
  Sample $G \sim p(G) \propto \exp\langle w, G\rangle$.
  $\mathcal{L} \leftarrow \mathbb{E}_{s,a\sim\mathcal{T}}[\ell(\pi_G(s), a)]$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(G, \mathcal{L})\}$
  Fit w on $\mathcal{D}$ with linear regression.
end for
Return: $\arg\max_G p(G)$

Policy execution mode

Algorithm 2 Policy execution intervention

Input: policy network f_ϕ s.t. $\pi_G(X) = f_\phi([X \odot G, G])$
Initialize $w = 0, \mathcal{D} = \emptyset$.
for $i = 1 \ldots N$ **do**
  Sample $G \sim p(G) \propto \exp\langle w, G\rangle$.
  Collect episode return R_G by executing π_G.
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(G, R_G)\}$
  Fit w on $\mathcal{D}$ with linear regression.
end for
Return: $\arg\max_G p(G)$

Disentangling Observations

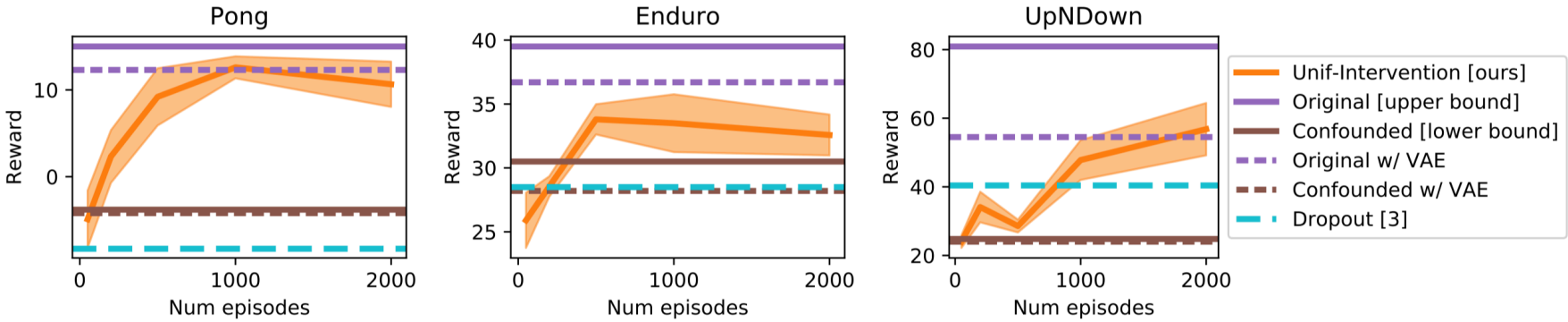


Figure 6: Reward vs. number of intervention episodes (policy execution interventions) on Atari games. UNIF-INTERVENTION succeeds in getting rewards close to ORIGINAL W/ VAE, while the DROPOUT baseline only outperforms CONFOUNDED W/ VAE in UpNDown.

Mode	Representation	Reward
Policy execution	Disentangled	**-137**
	Entangled	-145
Expert queries	Disentangled	**-140**
	Entangled	-165

Table 2: Intervention on (dis)entangled MountainCar.

Intervention by Policy Execution

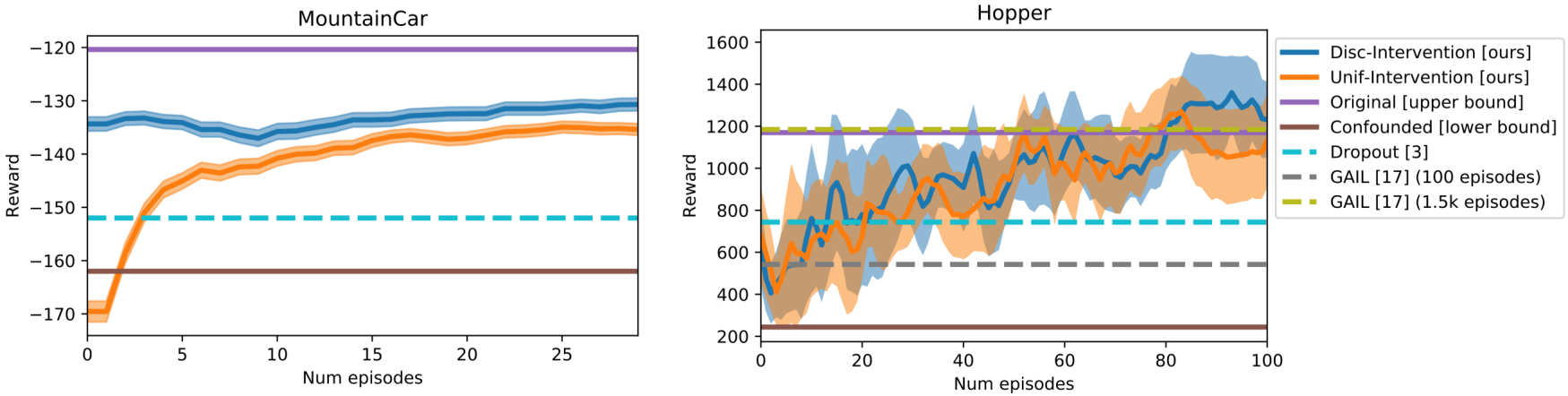


Figure 7: Reward vs. number of intervention episodes (policy execution interventions) on MountainCar and Hopper. Our methods UNIF-INTERVENTION and DISC-INTERVENTION bridge most of the causal misidentification gap (between ORIGINAL (lower bound) and CONFOUNDED (upper bound), approaching ORIGINAL performance after tens of episodes. GAIL [19] (on Hopper) achieves this too, but after 1.5k episodes.

Intervention by Expert Queries

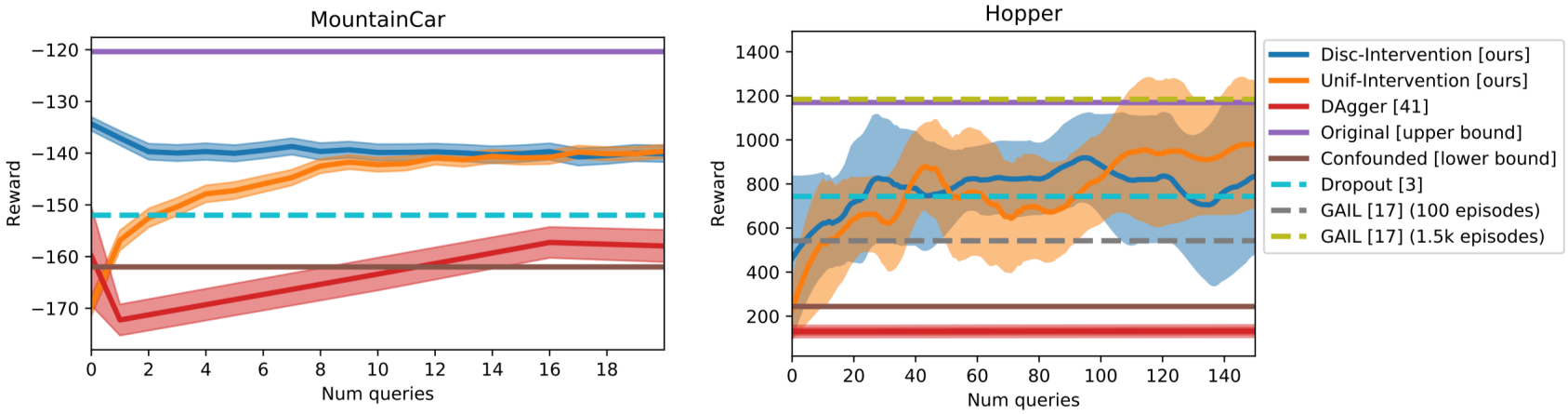


Figure 8: Reward vs. expert queries (expert query interventions) on MountainCar and Hopper. Our methods partially bridge the gap from CONFOUNDED (lower bd) to ORIGINAL (upper bd), also outperforming DAGGER [43] and DROPOUT [3]. GAIL [19] outperforms our methods on Hopper, but requires a large number of policy roll-outs (also see Fig 7 comparing GAIL to our policy execution-based approach).

Q & A

Outline