An Empirical Analysis of Measure-Valued Derivatives for Policy Gradients

João Carvalho[1], Davide Tateo[1], Fabio Muratore[1,2], Jan Peters[1,3]
Intelligent Autonomous Systems
Technische Universität Darmstadt, Darmstadt, Germany
{joao,davide,fabio}@robot-learning.de, jan.peters@tu-darmstadt.de

SUPPLEMENTARY MATERIAL

I. CODE TO REPLICATE THE EXPERIMENTS

The code is written in Python and makes use of PyTorch for automatic differentiation and the MushroomRL library for algorithm implementation and benchmarking https://github.com/MushroomRL/mushroom-rl. The code to reproduce the experiments is available at https://git.ias.informatik.tu-darmstadt.de/carvalho/mvd-rl.

II. OPTIMIZATION TEST FUNCTIONS

In Fig. 1 we maximize $\mathbb{E}_{p(\boldsymbol{x};\boldsymbol{\omega})}\left[f(\boldsymbol{x})\right]$ via gradient ascent, where f are three 2-dimensional test functions and $p(\boldsymbol{x};\boldsymbol{\omega})$ is a multivariate Gaussian distribution with diagonal covariance $p(\boldsymbol{x};\boldsymbol{\omega}) = \mathcal{N}\left(\boldsymbol{x};\boldsymbol{\omega} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}\right)$, with $\boldsymbol{x} \in \mathbb{R}^2$ and $\boldsymbol{\omega} \in \mathbb{R}^4$. The covariance is parameterized by the logarithm of standard deviation. The learning rate is kept constant and equal to 5×10^{-4} for all functions and estimators. The MVD uses one gradient estimate between parameter updates, while the SF and Rep-trick use the mean of eight estimates, which is the number of queries to f done with MVD for one estimate. The SF uses the optimal baseline for black-box optimization as computed by the PGPE algorithm.

Quadratic function
$f : \mathbb{R}^2 \to \mathbb{R}$
$f(\boldsymbol{x}) = -\boldsymbol{x}^\intercal \boldsymbol{x}$
Starting distribution: $\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu} = (-5,-5), \boldsymbol{\Sigma} = \text{diag}\left(2^2, 2^2\right)\right)$

Himmelblau function
$f : \mathbb{R}^2 \to \mathbb{R}$
$f(x, y) = -(x^2 + y - 11)^2 - (x + y^2 - 7)^2$
Starting distribution: $\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu} = (0,-6), \boldsymbol{\Sigma} = \text{diag}\left(2^2, 2^2\right)\right)$

Styblinski function
$f : \mathbb{R}^2 \to \mathbb{R}$
$f(\boldsymbol{x}) = -\frac{1}{2}\sum_{i=1}^{2}\left(x_i^4 - 16x_i^2 + 5x_i\right)$
Starting distribution: $\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu} = (0,0), \boldsymbol{\Sigma} = \text{diag}\left(2^2, 2^2\right)\right)$

III. LINEAR-QUADRATIC REGULATOR

The discounted infinite-horizon discrete-time LQR problem is defined as

$$\arg\max_{\boldsymbol{s}_t, \boldsymbol{a}_t} J = \sum_{t=0}^{\infty} -\gamma^t \left(\boldsymbol{s}_t^\intercal \mathbf{Q} \boldsymbol{s}_t + \boldsymbol{a}_t^\intercal \mathbf{R} \boldsymbol{a}_t\right)$$
$$\text{s.t. } \boldsymbol{s}_{t+1} = \mathbf{A}\boldsymbol{s}_t + \mathbf{B}\boldsymbol{a}_t.$$

The optimal control policy for this problem is a linear-in-the-state time-independent feedback controller $\boldsymbol{a}_t = -\mathbf{K}_{\text{opt}}\boldsymbol{s}_t$. In the experiments we compute the gradient w.r.t. $\mathbf{K}$ of a stochastic policy $\boldsymbol{a} \sim \mathcal{N}\left(\cdot| - \mathbf{K}\boldsymbol{s}_t, \boldsymbol{\Sigma}\right)$, with fixed diagonal covariance 0.1^2 in all action dimensions.

We build four environments with different dimensions of states and actions with $(|\mathcal{S}|, |\mathcal{A}|)$: $(2,1)$, $(2,2)$, $(4,4)$ and $(6,6)$. The matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{Q}$ and $\mathbf{R}$ can be found in the accompanying code. The dynamics matrices $\mathbf{A}$ and $\mathbf{B}$ are non-diagonal,

[1] Technische Universität Darmstadt, Darmstadt, Germany
[2] Honda Research Institute Europe, Offenbach am Main, Germany
[3] Max Planck Institute for Intelligent Systems, Tübingen, Germany

which makes the LQR more difficult to solve. The matrix $\mathbf{A}$ is chosen such that the system is unstable when $\boldsymbol{a}_t = 0$. The initial gain matrix $\mathbf{K}_{\text{init}}$ is chosen by sampling from a Gaussian distribution with mean $\mathbf{K}_{\text{opt}}$ and problem-dependent covariance, such that the closed-loop system is stable. The initial state for each environment is 9 for each dimension. For instance, in LQRs $(2,1)$ and $(2,2)$ the initial state is $\boldsymbol{s}_0 = (9,9)^\intercal$. The discount factor is $\gamma = 0.99$. To simulate infinite horizons, trajectory rollouts have $T = 1000$ steps. The policy gradients are computed from $\boldsymbol{s}_0$ and $\mathbf{K}_{\text{init}}$. The discounted state distribution is obtained by sampling trajectories and multiplying each state at time t with γ^t.

The experiments of Fig. 2 use the true Q and V functions computed with $\mathbf{K}_{\text{init}}$. Figures 3 and 4 simulate an error in the Q-function estimator with an added local sinusoidal term to the true state-action value function. This approximator is modelled as $\hat{Q}(\boldsymbol{s},\boldsymbol{a}) = Q(\boldsymbol{s},\boldsymbol{a}) + \alpha Q(\boldsymbol{s},\boldsymbol{a})\cos(2\pi f \boldsymbol{p}^\intercal \boldsymbol{a} + \phi)$. f is the error frequency, $\boldsymbol{p}$ is a random vector whose entries sum to 1, and $\phi \sim \mathcal{U}[0, 2\pi]$ a phase shift. $\boldsymbol{p}$ and ϕ introduce randomness to remove correlation between action dimensions. The factor α represents an error proportional to the true value, and the cosine term models adding a (high) frequency error component. These frequency components can appear in function approximators, especially if they overfit to the data. This is a simplified error model that uses only one frequency component, but it is useful to understand the sensitivity of the gradient estimators to local errors in function approximators.

Figures 9 and 10 complement the results of Fig. 2 with more trajectories, and a 6-dimensional LQR.

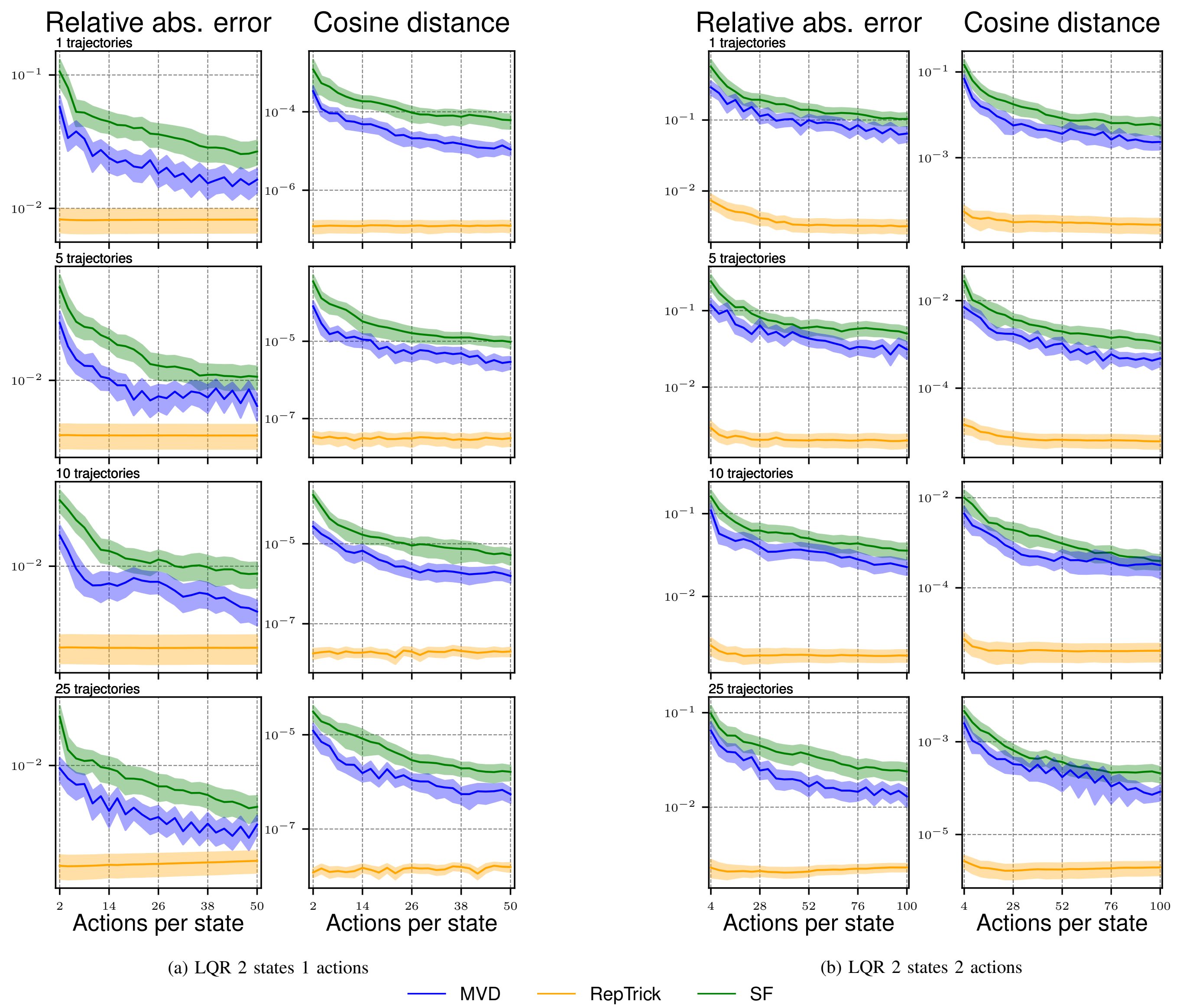
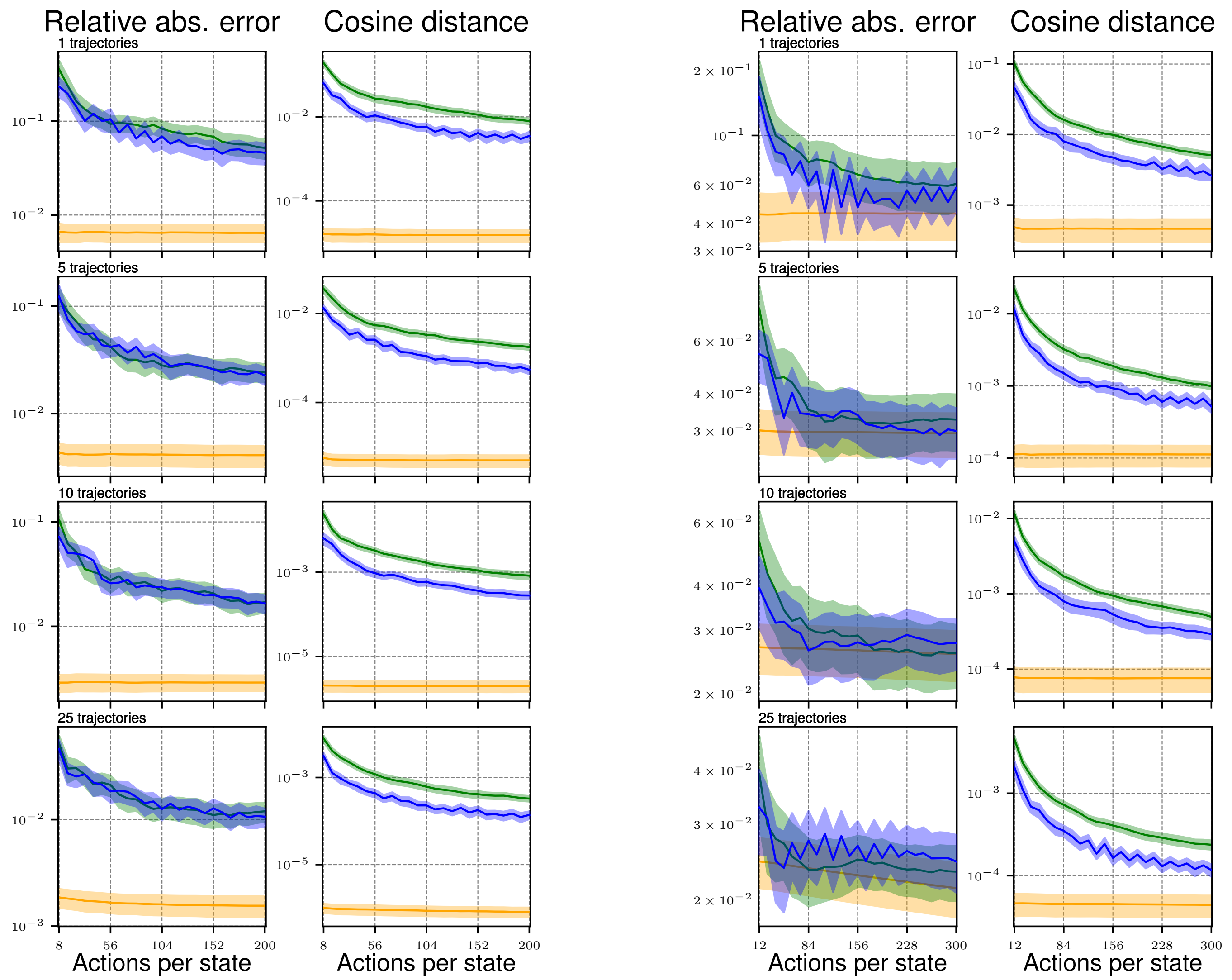


Fig. 9: Gradient errors in magnitude and direction in the LQRs (2 states, 1 action) and (2 states, 2 actions), per number of trajectories and sampled actions. Depicted are the mean and the 95% confidence interval of 25 random seeds.

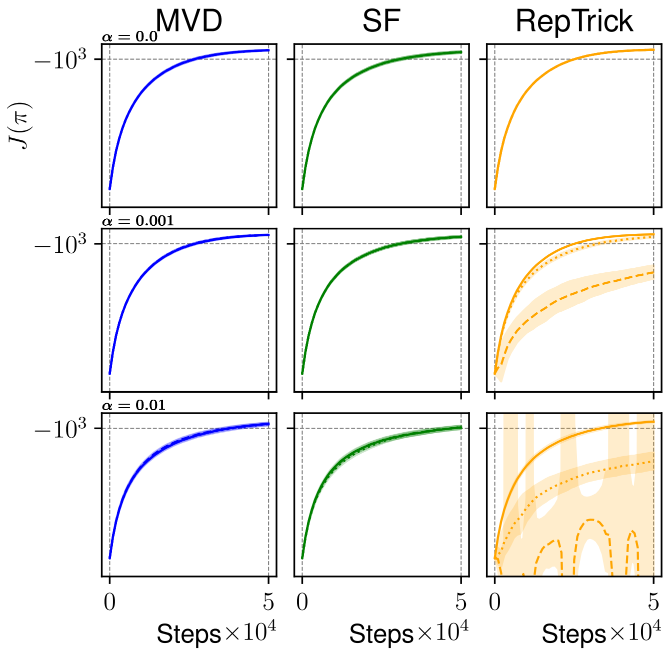


(a) LQR 4 states 4 actions

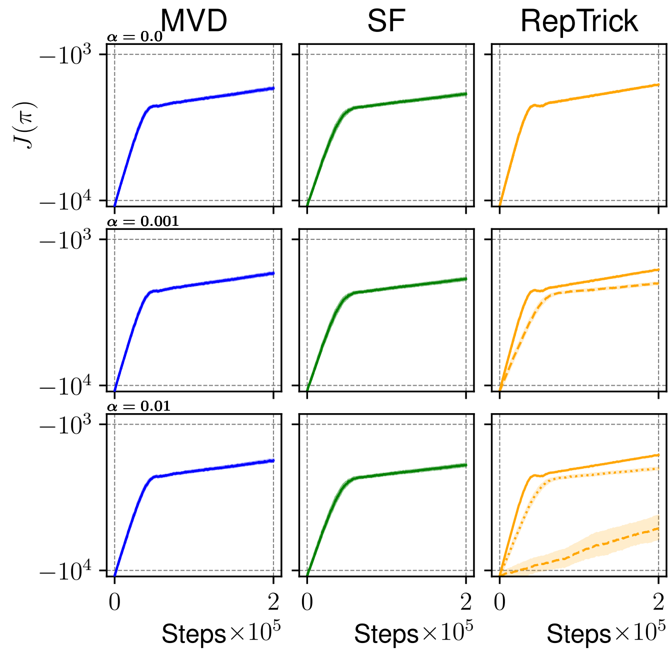
(b) LQR 6 states 6 actions

Fig. 10: Gradient errors in magnitude and direction in the LQRs (4 states, 4 actions) and (6 states, 6 actions), per number of trajectories and sampled actions. Depicted are the mean and the 95% confidence interval of 25 random seeds.

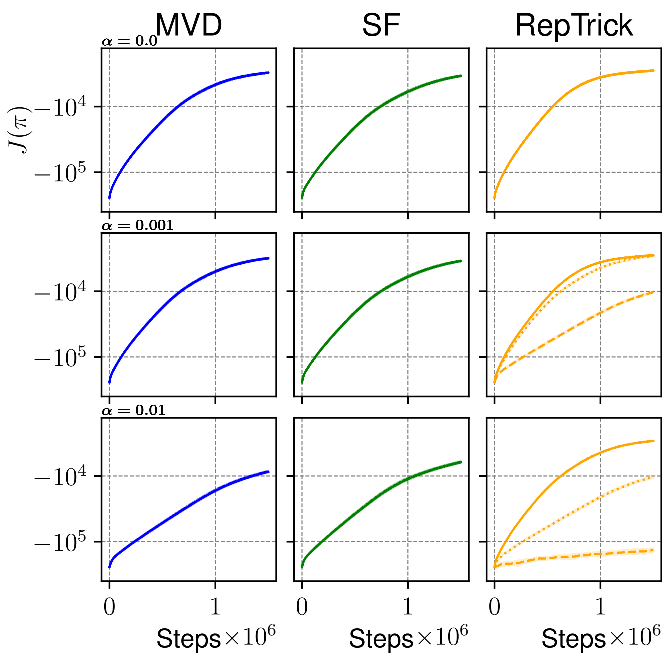
In Fig. 4 the initial policy is the same as in Fig. 2. The policy update uses the Adam optimizer and the following learning rates $(|\mathcal{S}|, |\mathcal{A}|)$: $(2, 1) : 5 \times 10^{-2}$; $(2, 2) : 1 \times 10^{-2}$; $(4, 4) : 3 \times 10^{-3}$; $(6, 6) : 5 \times 10^{-3}$. Fig. 11 complements the results without added noise ($\alpha = 0$), and a 6-dimensional LQR.



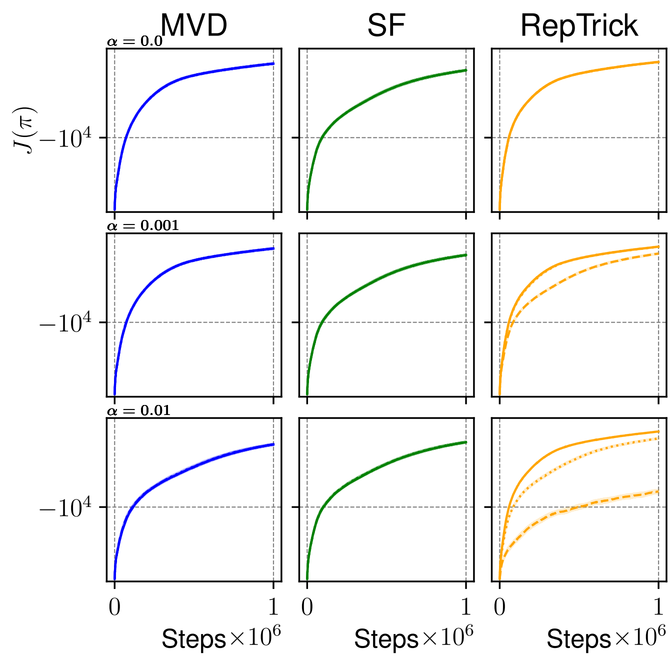
(a) LQR 2 states 1 actions



(b) LQR 2 states 2 actions



(c) LQR 4 states 4 actions



(d) LQR 6 states 6 actions

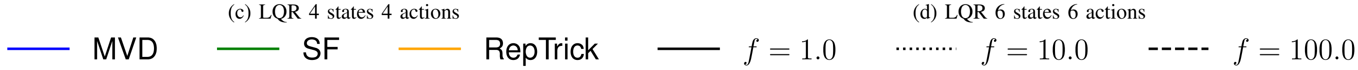


Fig. 11: Learning curves of the LQR tasks with an error in the Q-function approximator. Noise amplitudes (0.001, 0.01, 0.1, 1.0, 10.0), bottom to top.

IV. Off-Policy Experiments

Off-policy experiments from Fig. 5 use the environments from the PyBullet simulator [30]. Fig. 12 shows additional results in simpler tasks. Table III contain the used hyperparameters. The neural network architectures are from the original papers.

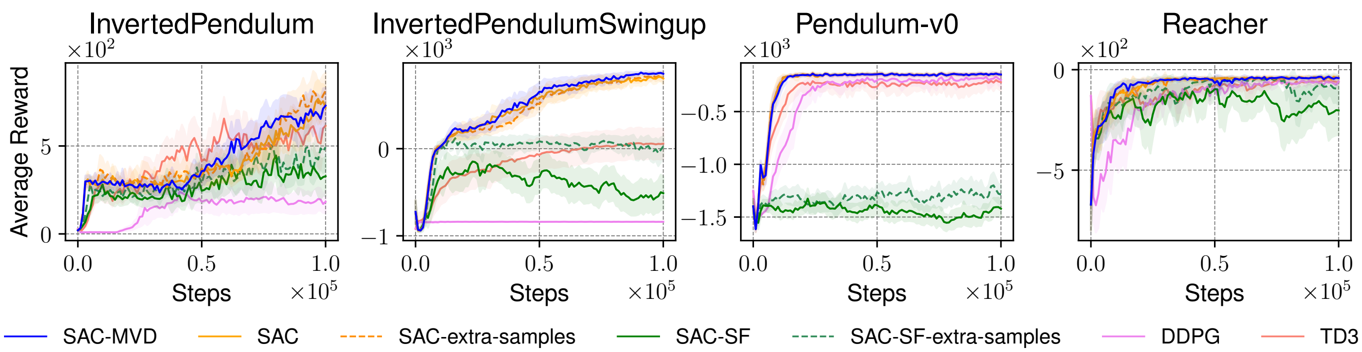


Fig. 12: Policy evaluation results per samples collected during training on different tasks in deep RL. Depicted are the average reward and the 95% confidence interval of 25 random seeds.

	Pendulum-v0	InvertedPendulum-v0 InvertedPendulumSwingup-v0 ReacherEnv-v0	Ant-v0, HalfCheetah-v0 Walker2d-v0, Hopper-v0
SAC variants			
horizon	200	1000	1000
γ	0.99	0.99	0.99
epochs	50	50	100
steps/epoch	1000	1000	10000
episodes evaluation	10	10	10
batch size	64	64	256
warmup transitions	128	128	10000
max replay size	500000	500000	500000
critic network	[64, 64] ReLU	[64, 64] ReLU	[256, 256] ReLU
actor network	[64, 64] ReLU	[64, 64] ReLU	[256, 256] ReLU
optimizer	Adam	Adam	Adam
lr actor	1×10^{-4}	1×10^{-4}	1×10^{-4}
lr critic	3×10^{-4}	3×10^{-4}	3×10^{-4}
DDPG and TD3			
batch size	64	64	256
warmup transitions	128	128	10000
max replay size	1000000	1000000	1000000
critic network	[64, 64] ReLU	[64, 64] ReLU	[400, 300] ReLU
actor network	[64, 64] ReLU	[64, 64] ReLU	[400, 300] ReLU
optimizer	Adam	Adam	Adam
lr actor	1×10^{-4}	1×10^{-4}	1×10^{-4}
lr critic	1×10^{-3}	1×10^{-3}	1×10^{-3}

TABLE III: Hyperparameters for the off-policy experiments.

V. On-Policy Experiments

For TRPO and PPO the policy is a Gaussian distribution with diagonal covariance, where the mean is the output of a neural network, and the log-standard deviation is a state independent learnable parameter, as in the original papers. Tree-MVD optimizes the same policy but applies a $\tanh$ operator to the sampled actions, as done in SAC. Applying this operator to MVDs is straight forward.

Tables IV and V contain the hyperparameters used in the experiments. The neural network architectures are taken from the original works.

	Pendulum-v0	LunarLanderContinuous-v2	Room	Corridor
horizon	200	1000	300	300
γ	0.99	0.99	0.99	
epochs	100	100	60	60
steps/epoch	3000	3000	2000	2000
episodes evaluation	10	10	10	10
iters bellman equation	100	50	25	25
tree estimators	100	50	25	25
min samples split node	2	2	8	8
min samples leaf node	1	1	4	4
max replay size	500000	500000	500000	500000
replay batch size	25000	25000	10000	10000
actor update epochs	4	4	4	4
actor batch size	256	128	128	128
actor network	[32, 32] ReLU	[32, 32] ReLU	[32, 32] ReLU	[32, 32] ReLU
optimizer	Adam	Adam	Adam	Adam
actor learning rate	3×10^{-4}	3×10^{-4}	1×10^{-4}	1×10^{-4}
initial σ	1	1	1	1

TABLE IV: Hyperparameters for the on-policy experiments with Tree-MVD.

	Pendulum-v0	LunarLanderContinuous-v2	Room	Corridor
horizon	200	1000	300	300
γ	0.99	0.99	0.99	
epochs	100	100	60	60
steps/epoch	3000	3000	2000	2000
episodes evaluation	10	10	10	10
critic update epochs	10	10	10	10
critic batch size	64	64	64	64
critic network	[32, 32] ReLU	[32, 32] ReLU	[128, 128] ReLU	[128, 128] ReLU
critic learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
actor update epochs	8	4	4	4
actor batch size	256	256	128	128
actor network	[32, 32] ReLU	[32, 32] ReLU	[32, 32] ReLU	[32, 32] ReLU
optimizer	Adam	Adam	Adam	Adam
actor learning rate	3×10^{-4}	3×10^{-4}	1×10^{-4}	1×10^{-4}
initial σ	1	1	1	1
PPO	$\epsilon = 0.2$, $\lambda(\text{GAE}) = 0.95$			
TRPO	$\max \text{KL} = 0.01$, $\lambda(\text{GAE}) = 0.95$, epochs line search $= 10$, epochs conj gradient $= 100$			

TABLE V: Hyperparameters for the on-policy experiments with PPO and TRPO.