General Policy Evaluation and Improvement by Learning to Identify Few But Crucial States

Francesco Faccio, Aditya Ramesh, Vincent Herrmann, Jean Harb and Jürgen Schmidhuber

{francesco,aditya,vincent.herrmann,juergen}@idsia.ch, jean.merheb-harb@mail.mcgill.ca

Problem and Motivation

- **Reinforcement Learning (RL):** Find optimal policy π^*
- **Policy optimization:** Given a class of policies, find the policy parameters maximizing $J(\pi_{\boldsymbol\theta})$ (Sutton et al., 1999):

$$J(\pi_{\boldsymbol\theta}) = \int_{\mathcal{S}} \mu_0(s) V^{\pi_{\boldsymbol\theta}}(s)\,\mathrm{d}s = \int_{\mathcal{S}} \mu_0(s) \int_{\mathcal{A}} \pi_{\boldsymbol\theta}(a|s) Q^{\pi_{\boldsymbol\theta}}(s,a)\,\mathrm{d}a\,\mathrm{d}s$$

- **Problem:** Value functions are defined for a *single policy*. During optimization, the information on previous policies is potentially lost
- **Solution idea:** Learn a single value function able to evaluate many policies

Parameter-Based Value Functions

- **Parameter-Based Value Functions** (Faccio et al., 2021) generalize over multiple policies by incorporating the policy parameters as an additional input
- **PSVF**: Parameter based state-value function

$$V(s, \boldsymbol\theta) := \mathbb{E}[R_t | s_t = s, \boldsymbol\theta]$$

- **PAVF**: Parameter based action-value function

$$Q(s, a, \boldsymbol\theta) := \mathbb{E}[R_t | s_t = s, a_t = a, \boldsymbol\theta]$$

- **PSSVF**: Parameter based start-state-value function

$$V(\boldsymbol\theta) := \mathbb{E}[R_0 | \boldsymbol\theta]$$

- The PSSVF models $J(\theta)$ directly as a differentiable function $V(\theta)$, which is the expectation of $V(s,\theta)$ over the initial states

$$V(\theta) := \mathbb{E}_{s\sim\mu_0(s)}[V(s,\theta)] = \int_{\mathcal{S}} \mu_0(s) V(s,\theta)\,\mathrm{d}s = J(\pi_\theta).$$

If we can learn $V_w(\theta)$, we can improve the policy by simply taking gradient ascent steps

$$\nabla_{\boldsymbol\theta} J(\pi_{\boldsymbol\theta}) = \nabla_{\boldsymbol\theta} V_w(\pi_{\boldsymbol\theta})$$

- **Problem:** How can we give the policy parameters as input to the value function when π_θ is a neural network?

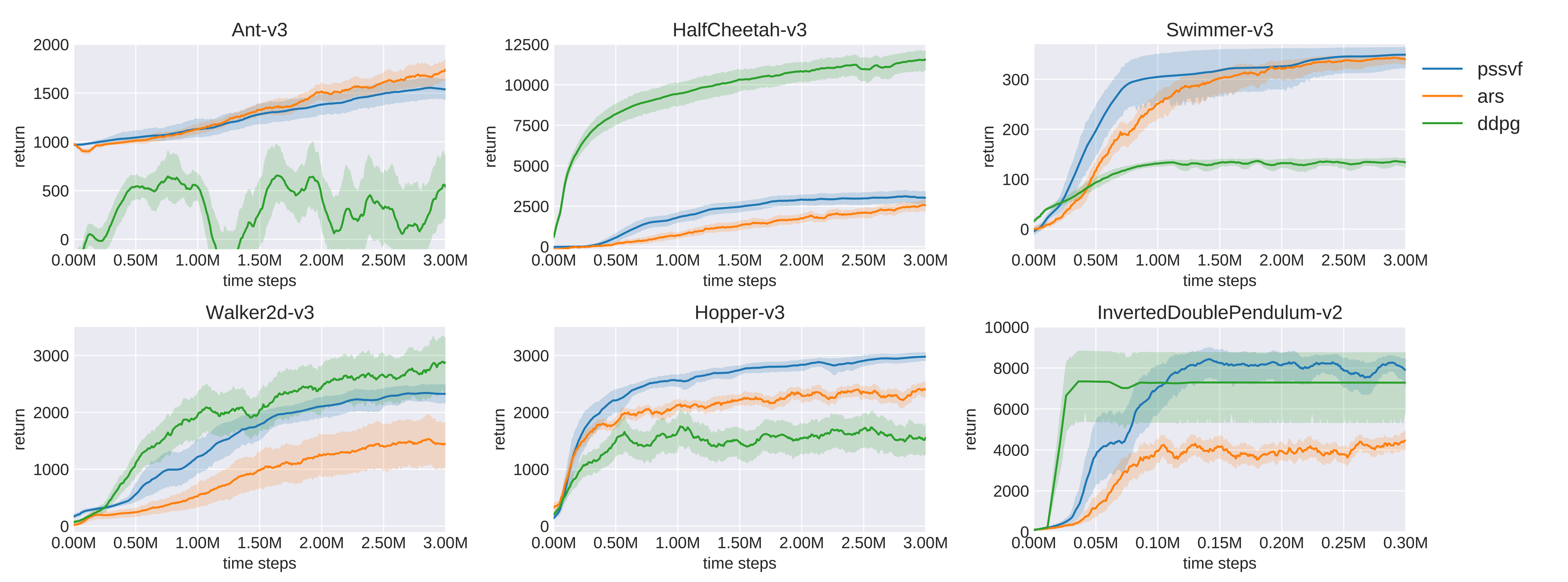
Policy fingerprinting

- **Policy fingerprinting** (Harb et al., 2020) creates a lower-dimensional policy representation
- It learns a set of K **probing states** $\{\tilde{s}_k\}_{k=1}^K$ and an evaluation function V_ϕ
- To evaluate a policy π_θ, it computes the 'probing actions' $\tilde{a}_k$ that the policy produces in the probing states. Then the concatenated vector of these actions is given as input to V_ϕ and mapped to the return
- Setting $w = \{\phi, \tilde{s}_1, \dots \tilde{s}_K\}$, we optimize V_w minimizing MSE:

$$\min_w \mathcal{L}_V := \min_w \mathop{\mathbb{E}}_{(\pi_\theta, r)\in B}[(V_w(\theta) - r)^2]$$
$$= \min_{\phi, \tilde{s}_1, \dots \tilde{s}_K} \mathop{\mathbb{E}}_{(\pi_\theta, r)\in B}[(V_\phi([\pi_\theta(\tilde{s}_1), \dots, \pi_\theta(\tilde{s}_K)]) - r)^2]$$

Main results

- Comparison with DDPG (Lillicrap et al., 2015) and ARS (Mania et al., 2018) using deep deterministic policies (2 hidden layers, 256 neurons per layer)



References

F. Faccio, L. Kirsch, and J. Schmidhuber. Parameter-based value functions. In *9th International Conference on Learning Representations, ICLR*, 2021.

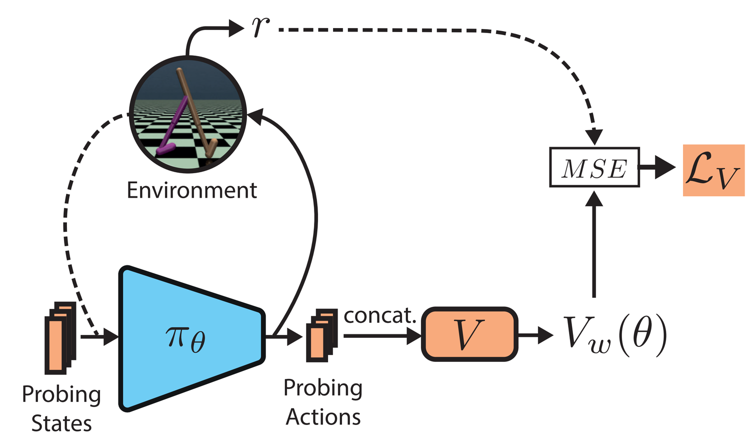
J. Harb, T. Schaul, D. Precup, and P.-L. Bacon. Policy evaluation networks. *arXiv preprint arXiv:2002.11833*, 2020.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

H. Mania, A. Guy, and B. Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems, NeurIPS*, 2018.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, 1999.

Actor Critic Algorithm



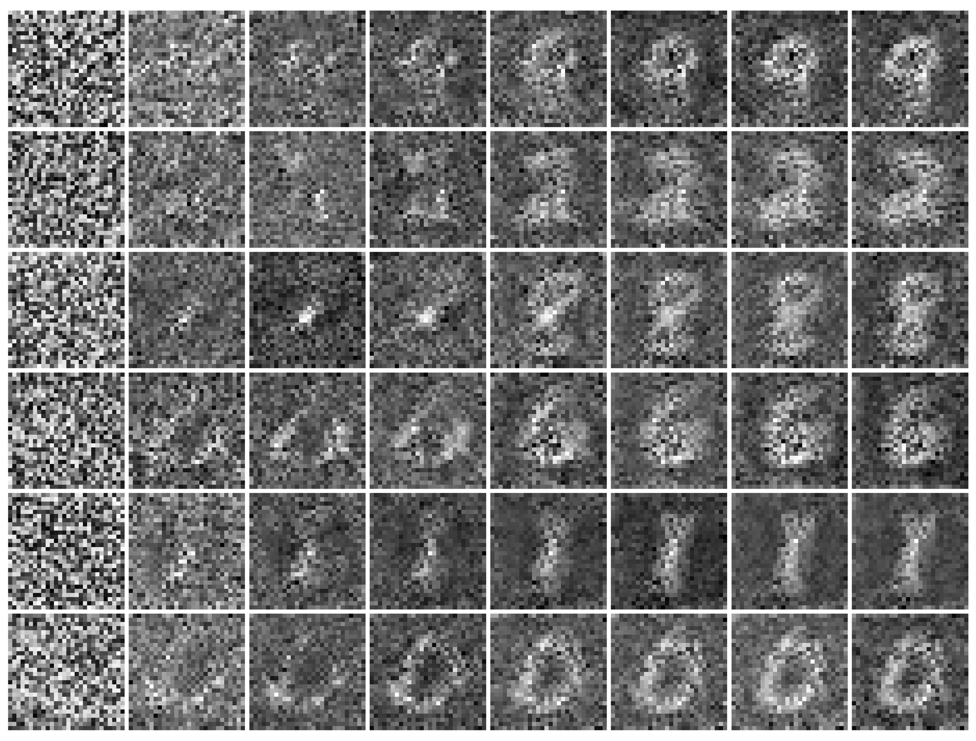
Off-policy actor-critic with PSSVF

Given the behavioral π_b, find $\pi_{\boldsymbol\theta}$ **maximizing** $J(\boldsymbol\theta)$:

1. Collect data with π_b (expensive in RL)
2. Use data to train $V(\boldsymbol\theta)$
3. Improve $\pi_{\boldsymbol\theta}$ following $\nabla_{\boldsymbol\theta} J(\pi_{\boldsymbol\theta})$ (offline optimization)
4. Set new behavioral $\pi_{\boldsymbol\theta} \leftarrow \pi_b$
5. Repeat until convergence

Demonstration on MNIST

- We apply our algorithm to MNIST classification. The return is the negative supervised loss. We obtain 87% accuracy. Learned probing states are digits

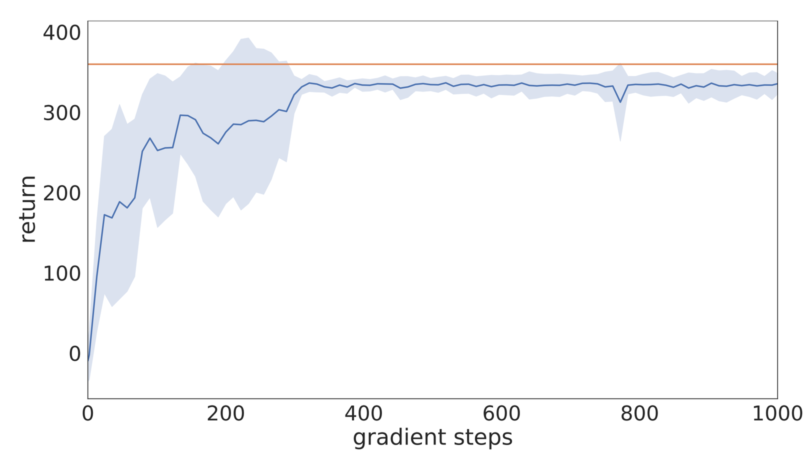


Offline policy improvement:

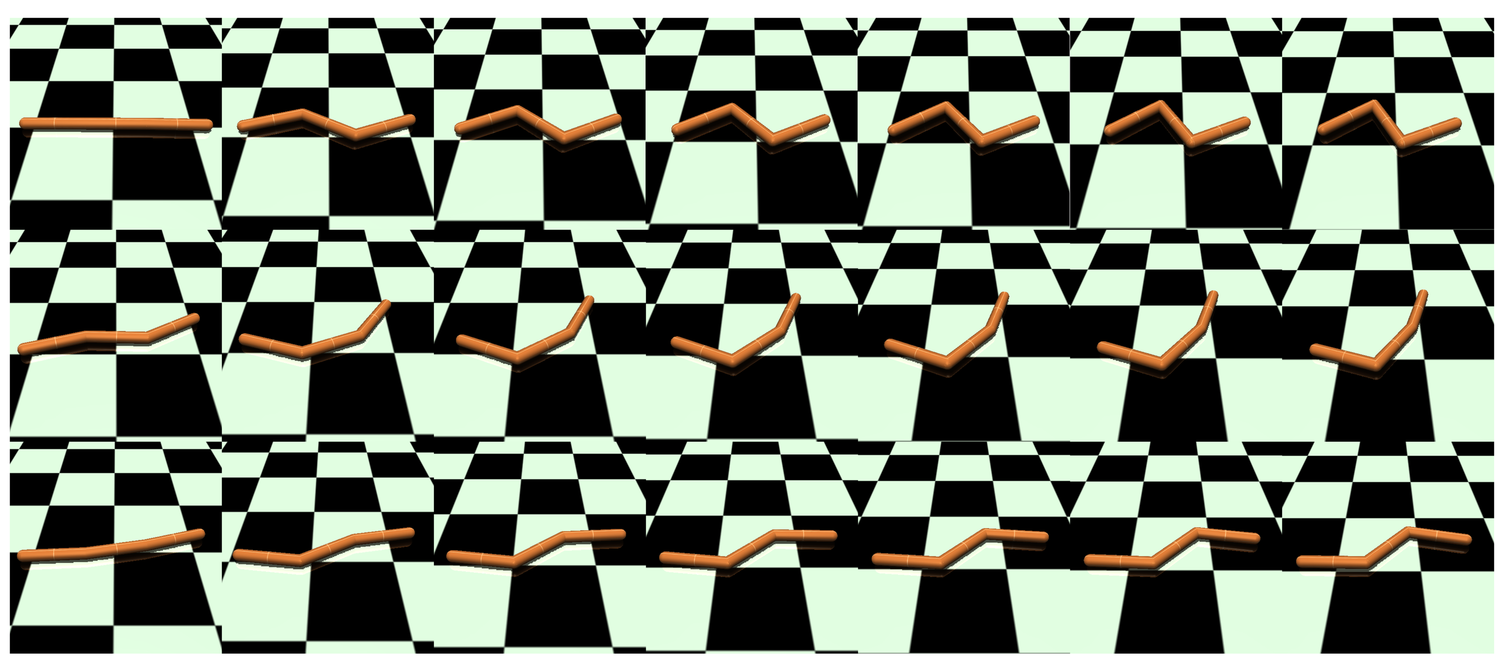
- With an offline dataset $\{\pi_{\theta_i}, l_i\}_{i=1}^N$ of randomly initialized CNNs and their losses (maximum accuracy 12%), we train V_w to evaluate such policies
- We randomly initialize a new CNN and take many steps of gradient ascent through the fixed value function, obtaining a final CNN whose accuracy is around 65% on the test set

More Results

- A PSSVF trained using deep deterministic policies zero-shot learns a linear policy with similar performance in Swimmer



- The method extracts crucial abstract knowledge about the environment in form of very few learned abstract states sufficient to fully specify the behavior of many policies
- A randomly initialized policy can learn optimal behaviors in Swimmer by knowing how to act only in the following 3 crucial learned states (similar results for Hopper with 5 states):



- Videos of learned probing states for the RL experiments: