A MORE CONTENT ON POLICY ABSTRACTION THEORY

A.1 Other Policy Abstractions

In this paper, we also propose three other policy abstractions in Definition 5. Before introducing the definitions of additional policy abstractions, we make some necessary notations. We use $d^{\pi,k}(\cdot)$ to denote the distribution of state when policy π performs k steps from initial states regarding the initial state distribution ρ_0. The discounted state visitation distribution from initial states regarding ρ_0 is defined as $d^{\pi}(s') = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t d^{\pi,t}(s')$ for any $s' \in S$. Additionally, we use $d_s^{\pi,k}(\cdot)$ to denote the distribution of state when policy π performs k steps from any state s. The discounted state visitation distribution from any state s is defined as $d_s^{\pi}(s') = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t d_s^{\pi,t}(s')$ for any $s' \in S$.

Definition 5. Given an MDP $\langle S, A, P, R, \gamma\rangle$ and a ground policy space Π, for any two policies $\pi_i, \pi_j \in \Pi$, we define three more policy abstractions as follows:

1. An influence-irrelevance abstraction ($f_{d_s^{\pi}}$) is such that for all $s, s' \in S$, $f_{d_s^{\pi}}(\pi_i) = f_{d_s^{\pi}}(\pi_j)$ implies that $d_s^{\pi_i}(s') = d_s^{\pi_j}(s')$.
2. An influence-irrelevance abstraction ($f_{d^{\pi}}$) is such that for all $s \in S$, $f_{d^{\pi}}(\pi_i) = f_{d^{\pi}}(\pi_j)$ implies that $d^{\pi_i}(s) = d^{\pi_j}(s)$.
3. A value-irrelevance abstraction ($f_{J^{\pi}}$) is such that $f_{J^{\pi}}(\pi_i) = f_{J^{\pi}}(\pi_j)$ implies that $\mathbb{E}_{s_0\sim\rho_0}\left[V^{\pi_i}(s_0)\right] = \mathbb{E}_{s_0\sim\rho_0}\left[V^{\pi_j}(s_0)\right]$.

Similarly, we prove how the newly proposed policy abstractions are related to other abstractions we introduce in the main body of the paper.

Theorem 2 (Partial Ordering ($\succeq$))**.** Under the Definition 2, 3 and 5, if the reward function R depends only on state $s \in S$, we have ❶ $f_{\pi} \succeq f_{P^{\pi}} \succeq f_{d_s^{\pi}} \succeq f_{d^{\pi}} \succeq f_{J^{\pi}}$; ❷ $f_{\pi} \succeq f_{P^{\pi}} \succeq f_{d_s^{\pi}} \succeq f_{V^{\pi}} \succeq f_{J^{\pi}}$. An illustration is provided below:

$$\begin{array}{ccccccc} f_{\pi} & \succeq & f_{P^{\pi}} & \succeq & f_{d_s^{\pi}} & \succeq & f_{d^{\pi}} \\ & & & & \curlyvee & & \curlyvee \\ & & & & f_{V^{\pi}} & \succeq & f_{J^{\pi}} \end{array}$$

Proof. The partial ordering ($\succeq$) satisfies transitivity, thus, let us prove the Theorem 2 one by one in the following.

❶$f_{P^{\pi}} \succeq f_{d_s^{\pi}}$. Given an MDP M, two policies $\pi_i, \pi_j \in \Pi$. If $f_{P^{\pi}}(\pi_i) = f_{P^{\pi}}(\pi_j)$, with the Definition 2, we have $\forall s, s' \in S, P^{\pi_i}(s' \mid s) = P^{\pi_j}(s' \mid s)$. Since the initial state distribution are the same, and π_i, π_j follow two identical Markov chains, we derive $f_{P^{\pi}} \succeq f_{d_s^{\pi}}$.

❷$f_{d_s^{\pi}} \succeq f_{d^{\pi}}$. Given an MDP, two policies $\pi_i, \pi_j \in \Pi$. If $f_{d_s^{\pi}}(\pi_i) = f_{d_s^{\pi}}(\pi_j)$, with the Definition 5, we have $\forall s, s' \in S$, $d_s^{\pi_i}(s') = d_s^{\pi_j}(s')$. Furthermore, we derive,

$$\forall s_0 \in \rho_0, s' \in S,\ d_{s_0}^{\pi_i}(s') = d_{s_0}^{\pi_j}(s'). \tag{10}$$

Thus, $f_{d_s^{\pi}} \succeq f_{d^{\pi}}$.

❸$f_{d^{\pi}} \succeq f_{J^{\pi}}$. Given an MDP, two policies $\pi_i, \pi_j \in \Pi$. If $f_{d^{\pi}}(\pi_i) = f_{d^{\pi}}(\pi_j)$, with the Definition 5, we have $\forall s \in S$, $d^{\pi_i}(s) = d^{\pi_j}(s)$. When the reward function R depends only on state $s \in S$, we have,

$$\begin{aligned} J(\pi_i) &= (1-\gamma)^{-1}\mathbb{E}_{s\in d^{\pi_i}}[R(s)], \\ J(\pi_j) &= (1-\gamma)^{-1}\mathbb{E}_{s\in d^{\pi_j}}[R(s)]. \end{aligned} \tag{11}$$

Thus, $f_{d^{\pi}} \succeq f_{J^{\pi}}$.

❹$f_{d_s^{\pi}} \succeq f_{V^{\pi}}$. Given an MDP, two policies $\pi_i, \pi_j \in \Pi$. If $f_{d_s^{\pi}}(\pi_i) = f_{d_s^{\pi}}(\pi_j)$, with the Definition 5, we have $\forall s, s' \in S$, $d_s^{\pi_i}(s') = d_s^{\pi_j}(s')$. When the reward function R depends only on state $s' \in S$, we have,

$$\begin{aligned} \forall s \in S,\ V^{\pi_i}(s) &= (1-\gamma)^{-1}\mathbb{E}_{s'\in d_s^{\pi_i}}[R(s')], \\ V^{\pi_j}(s) &= (1-\gamma)^{-1}\mathbb{E}_{s'\in d_s^{\pi_j}}[R(s')]. \end{aligned} \tag{12}$$

Thus, $f_{d_s^{\pi}} \succeq f_{V^{\pi}}$.

❺$f_{V^{\pi}} \succeq f_{J^{\pi}}$. Given an MDP, two policies $\pi_i, \pi_j \in \Pi$. If $f_{V^{\pi}}(\pi_i) = f_{V^{\pi}}(\pi_j)$, with the Definition 2, we have $\forall s \in S, V^{\pi_i}(s) = V^{\pi_j}(s)$. Furthermore, we derive,

$$\begin{aligned} &\forall s_0 \in \rho_0,\ V^{\pi_i}(s_0) = V^{\pi_j}(s_0), \\ &\mathbb{E}_{s_0\sim\rho_0}\left[V^{\pi_i}(s_0)\right] = \mathbb{E}_{s_0\sim\rho_0}\left[V^{\pi_j}(s_0)\right]. \end{aligned} \tag{13}$$

Thus, $f_{V^{\pi}} \succeq f_{J^{\pi}}$.

In particular, when the reward function R depends on both state $s \in S$ and action $a \in A$, π_i may not be equivalent to π_j. The $f_{d^{\pi}} \succeq f_{J^{\pi}}$ and $f_{d_s^{\pi}} \succeq f_{V^{\pi}}$ are not hold. Connecting Definition 2,5 and Theorem 2, we summarize the properties of different policy abstractions in Tab. 1. □

B MORE CONTENT ON POLICY METRIC

Definition 6 (Metrics [23])**.** Let X be a non-empty set of data elements and a **metric** is a real-valued function $d\colon X \times X \to [0, \infty)$ such that for all $x, y, z \in X$:

(1) $d(x, y) = 0 \iff x = y$;
(2) $d(x, y) = d(y, x)$;
(3) $d(x, y) \le d(x, z) + d(z, y)$.

A **pseudo-metric** d is a metric with the first condition replaced by $x = y \implies d(x, y) = 0$. The combination $\langle X, d\rangle$ is called a metric space.

B.1 Estimating Policy Metrics via Jeffrey Divergence

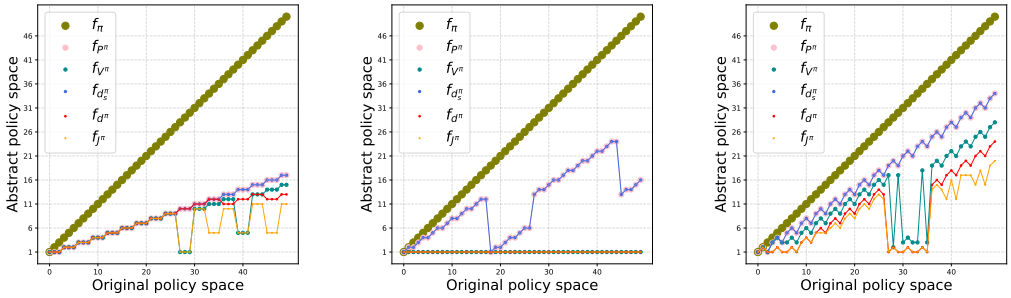
In simple MDPs where the state-action space is finite, we calculate the frequency distribution (i.e., $\widetilde{\pi}$, $\widetilde{P^{\pi}}$, $\widetilde{Z^{\pi}}$) using sufficient samples as an estimate of the exact probability distribution (i.e., π, P^{π}, Z^{π}). Then we use the Jeffreys Divergence [12] between empirical distributions as policy metrics (i.e., $d_{\pi}(\cdot,\cdot)$, $d_{P^{\pi}}(\cdot,\cdot)$, and $d_{V^{\pi}}(\cdot,\cdot)$).

$$\begin{aligned} d_{\pi}(\pi_i, \pi_j) &\approx D_{KL}\left(\widetilde{\pi}_i \| \widetilde{\pi}_j\right) + D_{KL}\left(\widetilde{\pi}_j \| \widetilde{\pi}_i\right), \\ d_{P^{\pi}}(\pi_i, \pi_j) &\approx D_{KL}\left(\widetilde{P}^{\pi_i} \| \widetilde{P}^{\pi_j}\right) + D_{KL}\left(\widetilde{P}^{\pi_j} \| \widetilde{P}^{\pi_i}\right), \\ d_{V^{\pi}}(\pi_i, \pi_j) &\approx D_{KL}\left(\widetilde{Z}^{\pi_i} \| \widetilde{Z}^{\pi_j}\right) + D_{KL}\left(\widetilde{Z}^{\pi_j} \| \widetilde{Z}^{\pi_i}\right). \end{aligned} \tag{14}$$

C MORE CONTENT ON EXPERIMENTS

C.1 Experimental Details of Policy Abstraction Theory

Experimental Setting. The Nchain is implemented with two loops that cover nine states and involve two actions, a and b. The agent receives a reward of 1 and 2 when transitioning to state 0 from state 4 and 8, respectively, and the reward for the remaining transitions is 0. With a slip probability of 0.2, the agent performs the opposite action. The Upworld is a 3×3 grid, which consists of nine states



(a) Nchain (b) Upworld (c) Random

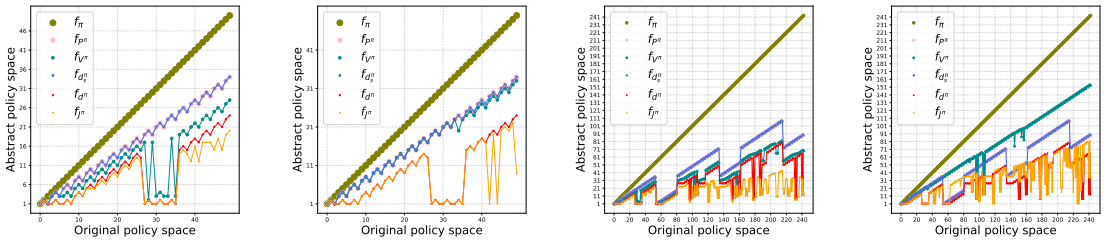
Figure 5: The illustration of different policy abstractions in three example domains. *(a)* Nchain; *(b)* Upworld; *(c)* Random. For each domain, the x-axis shows the top 50 policies in the ground policy space, and the y-axis shows the corresponding abstract policy in the abstract policy space for each policy in the ground policy space based on different policy abstractions.

Table 5: Comparison of policy space size based on different policy abstractions across domains (Nchain, Upworld, Random). $\zeta(\Pi)$ and $\zeta(f_*)$ denote the original policy space size and abstract policy space size corresponding to six policy abstractions.

Domain	Policy Space						
	$\zeta(\Pi)$	$\zeta(f_\pi)$	$\zeta(f_{P^\pi})$	$\zeta(f_{d_s{}^\pi})$	$\zeta(f_{d^\pi})$	$\zeta(f_{V^\pi})$	$\zeta(f_{J^\pi})$
Nchain	**19683**	**19683**	**81**	**81**	**69**	**63**	**33**
Upworld	**19683**	**19683**	**8748**	**8748**	**61**	**161**	**6**
Random	**243**	**243**	**108**	**108**	**80**	**82**	**44**

Table 6: Comparison of policy space size based on different policy abstractions for Random domain. $\zeta(\Pi)$ and $\zeta(f_*)$ denote the original policy space size and abstract policy space size corresponding to six policy abstractions.

Reward	Policy Space						
	$\zeta(\Pi)$	$\zeta(f_\pi)$	$\zeta(f_{P^\pi})$	$\zeta(f_{d_s{}^\pi})$	$\zeta(f_{d^\pi})$	$\zeta(f_{V^\pi})$	$\zeta(f_{J^\pi})$
R(s)	**243**	**243**	**108**	**108**	**80**	**82**	**44**
R(s,a)	**243**	**243**	**108**	**108**	**80**	**154**	**82**



(a) (b) (c) (d)

Figure 6: The illustration of different policy abstractions in the Random domain. *(a)* Random(R(s)/$\hat{\Pi}$); *(b)* Random(R(s,a)/$\hat{\Pi}$); *(c)* Random(R(s)/Π); *(d)* Random(R(s,a)/Π). The x-axis shows the ground policy space and the y-axis shows the corresponding abstract policy space based on different policy abstractions.

and three actions a, b, and c. The agent obtains a reward of 10 for transitions between states at the top of the grid, and the reward for the remaining transitions is 0. We implement the Random MDP with 5 states and 3 actions. In each state, each action transitions to one of two randomly selected states with a probability of 0.5. The reward function $R(s_t)$ is randomly initialized. In this study, we obtain the ground policy space of the three domains using the stochastic policies for Nchain and the deterministic policies for Upworld and Random MDP. Specifically, each policy in Nchain selects each action with a probability of 0, 1, or 0.5.

Experimental Results. Tab. 5 and Fig. 5 compare policy abstraction in two aspects: the size of the abstract policy space and the results of policy aggregation. Clearly, the experimental results indicate the

partial ordering (❶ $f_\pi \succeq f_{P^\pi} \succeq f_{d_s^\pi} \succeq f_{d^\pi} \succeq f_{J^\pi}$; ❷ $f_\pi \succeq f_{P^\pi} \succeq f_{d_s^\pi} \succeq f_{V^\pi} \succeq f_{J^\pi}$) between policy abstractions. The coarser the abstraction is, the more the original policy space is abstracted. In addition, to compare the two cases regarding the reward function, we consider two cases for the reward function in Random MDP, i.e., $R(s_t)$ and $R(s_t, a_t)$, which are randomly initialized. Tab. 6 and Fig. 6 report the experimental results of the two cases. Obviously, if the reward is independent of the dynamics caused by s, a, the $f_{d^\pi} \succeq f_{J^\pi}$ and $f_{d_s^\pi} \succeq f_{V^\pi}$ will not be held. As mentioned earlier, such a case is a minority in the ones of interest and has little impact on selecting an appropriate policy abstraction for a specific downstream problem. In Fig. **??**, we present the abstracted results of ground full policy space based on different levels of abstraction. In particular, we omit the distribution-irrelevance abstraction since $\zeta(\Pi) = \zeta(f_\pi)$. The abstraction results of the full policy space are consistent with the partial ordering of policy abstractions.

C.2 Experimental Details of Policy Metric

Experimental Setting. To quantitatively compare these policy metrics, we demonstrate how the distances of two policies (blue and green in Fig. 7) measured by the corresponding policy metrics differ in several Gridworld MDPs. We borrow *Distinct Policies*, *Doorway* from [13] and design a new environment, *Key Action* for simple prototypes of environments with different features. All three environments are *5×5* Gridworld, with the starting state and goal in the lower left and upper right grid, respectively. The discrete action space is {*up, down, left, right*}. We obtain the estimated value $v(\cdot)$ of each state by estimating 1 - `[expected number of step to goal when starting from a given state]`. About the *Key Action*, the agent obtain a positive reward only if it chooses *right* at the key state (i.e., marked by the red box in Fig. 7). Every agent rollouts 10k episodes for comparing different policy metrics. To be specific, a policy is represented by tensor $\pi \in \mathbb{R}^{5\times5\times4}$ and the distribution-irrelevance metric is defined as $d_\pi(\pi, \hat{\pi}) = \mathbb{E}_{\pi_{i,j,k}\in\pi, \hat{\pi}_{i,j,k}\in\hat{\pi}}\left[\left|\pi_{i,j,k} - \hat{\pi}_{i,j,k}\right|\right]$. Similarly, the state transition dynamics induced by policy is represented by tensor $P^\pi \in \mathbb{R}^{5\times5\times5}$ and the influence-irrelevance metric is defined as $d_{P^\pi}(\pi, \hat{\pi}) = \mathbb{E}_{P^\pi_{i,j,k}\in P^\pi, P^{\hat{\pi}}_{i,j,k}\in P^{\hat{\pi}}}\left[\left|P^\pi_{i,j,k} - P^{\hat{\pi}}_{i,j,k}\right|\right]$. Different from the d_{P^π} sampling-based estimation, we leverage the dynamic programming to learn the value function $V^\pi \in \mathbb{R}^{5\times5}$. Thus, the value-irrelevance metric is defined as $d_{V^\pi}(\pi, \hat{\pi}) = \mathbb{E}_{V^\pi_{i,j}\in V^\pi, V^{\hat{\pi}}_{i,j}\in V^{\hat{\pi}}}\left[\left|V^\pi_{i,j} - V^{\hat{\pi}}_{i,j}\right|\right]$. In addition, we also compare the two other policy abstractions proposed in Appendix A.1. Among them, we estimate the discounted state visitation distribution $d^\pi(s)$ via dividing visits to a state s by a total number of interactions (i.e., $d_{d^\pi}(\pi, \hat{\pi}) = \mathbb{E}_{d^\pi_{i,j}\in d^\pi, d^{\hat{\pi}}_{i,j}\in d^{\hat{\pi}}}\left[\left|d^\pi_{i,j} - d^{\hat{\pi}}_{i,j}\right|\right]$). Simply and naturally, based on the definition of f_{J^π} in Definition 5, we have $d_{J^\pi}(\pi, \hat{\pi}) = \left|\mathbb{E}_{s_0\sim\rho_0}\left[V^\pi(s_0)\right] - \mathbb{E}_{s_0\sim\rho_0}\left[V^{\hat{\pi}}(s_0)\right]\right|$.

Experimental Results. Fig. 7 shows the illustrations and the results of the five policy metrics. The d_{d^π} and d_{J^π} are newly added results compared to the original paper. We observe that the d_{d^π} cannot indicate the difference in dynamics between the two policies in *Key Action*. Like the d_{V^π}, the d_{J^π} measures the difference in the outcomes of the two policies but shows the poor robustness as the increase of stochasticity.

Table 7: Training details for our method. Grid search determines the best hyperparameter configuration for terms with multiple alternatives (i.e., {}).

Hyperparameters	Value
Actor Epoch	10
Critic Epoch	10
Policy Learning Rate	10^{-4}
Value Learning Rate	10^{-3}
Optimizer	Adam
Clipping Range Parameter (ϵ)	0.2
GAE Parameter (λ)	0.95
Discount Factor (γ)	0.99
On-policy Samples Per Iteration (n)	5 episodes or 2000 time steps
Update Interval (U)	Every 10 time steps
Batch Size	{64, 128}
Experience Buffer Size ($\mathbb{D}$)	200k (steps)
Policy Num Per Batch	{16, 32}
AL Learning Rate	10^{-3}
Gaussian Kernel (kernel_mul)	2.0
Gaussian Kernel (kernel_num)	5
AL Sample Pair Num (N, M)	1000
AL Scaling Weight η	{0.5, 1, 2}

C.3 Experimental Details of Policy Representation Method

Experimental Setting. Our experiments in this subsection are conducted in commonly adopted OpenAI Gym[1] continuous control tasks, i.e., *LunarLander* from Box2D and *HalfCheetah*, *Hopper*, *Walker2d*, *Ant* and *InvertedDoublePendulum* from MuJoCo [29]. We use the OpenAI Gym with version 0.9.1, the mujoco-py with version 0.5.4 and the MuJoCo products with version MJPRO131. Our codes are implemented with Python 3.6 and Tensorflow. We use Proximal Policy Optimization (PPO)[25] with Generalized Advantage Estimator (GAE) as our baseline lgorithm. For a fair comparison and clear evaluation, we perform no code-level optimization in our experiments, e.g., state standardization, reward scaling, gradient clipping, parameter sharing, etc. The algorithm PPO-PeVFA[28] is implemented based on PPO[25], which only differs in the replacement of the conventional value function network with the PeVFA network.

[1] http://gym.openai.com/

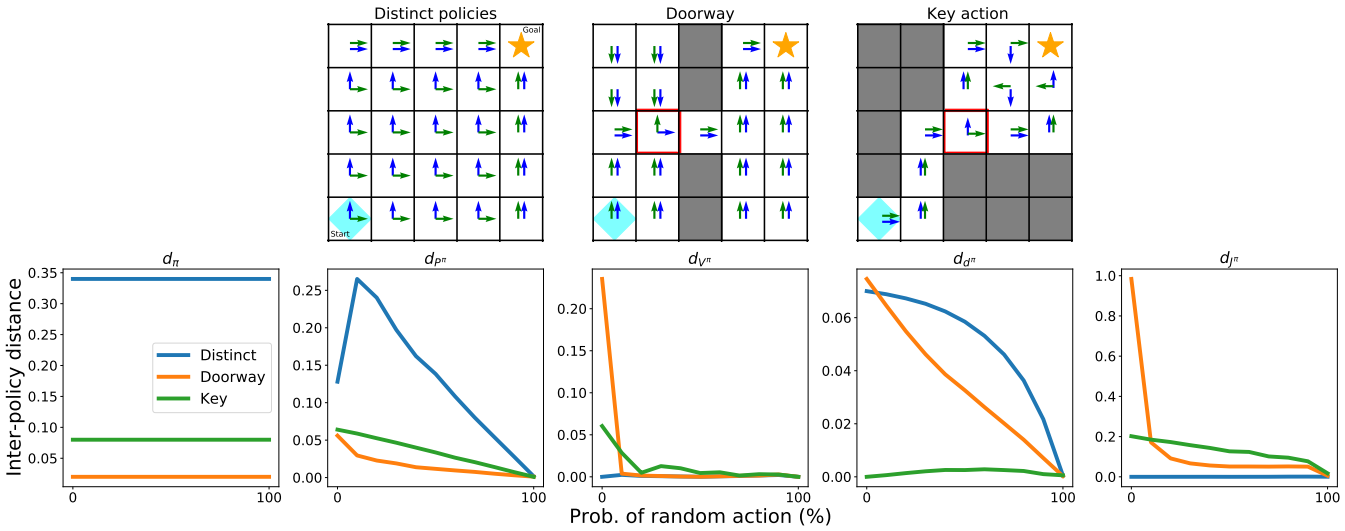


Figure 7: Policy comparison with different policy metrics in Gridworld. *Top Panel:* The illustration of three Gridworld MDPs and two deterministic policies (blue and green). *Bottom Panel:* The distance curves of the two policies measured by $d_\pi, d_{P^\pi}, d_{V^\pi}, d_{d^\pi}, d_{J^\pi}$ (y-axi), against the stochasticity of the environment (x-axi).

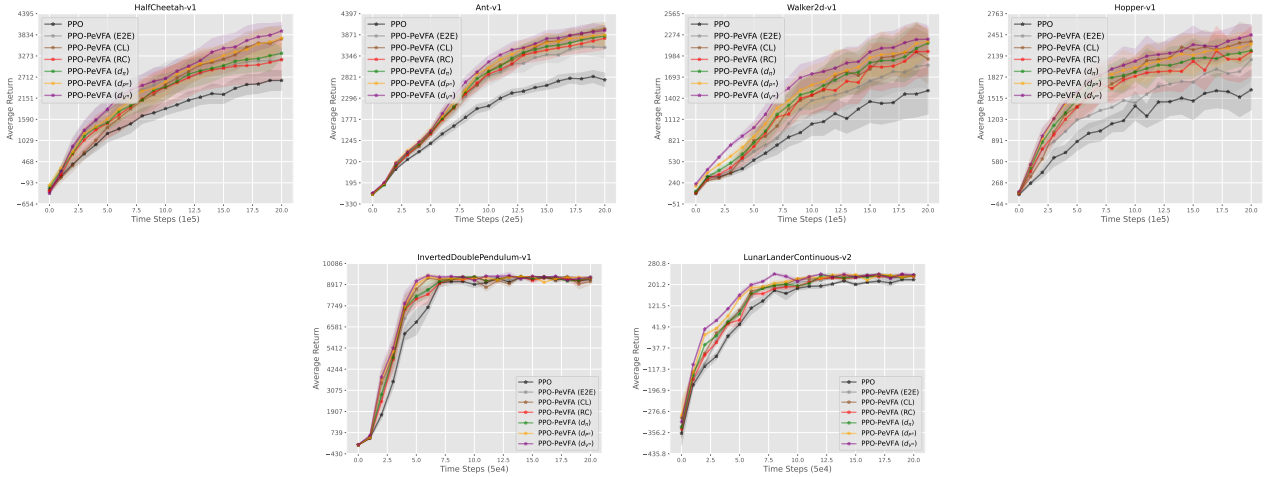


Figure 8: Learning curves for the OpenAI gym continuous control tasks. The shaded region represents half a standard deviation of the average evaluation over 10 trials. Curves are smoothed uniformly for visual clarity.