Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables

Kate Rakelly [1 *] **Aurick Zhou** [1 *] **Deirdre Quillen** [1] **Chelsea Finn** [1] **Sergey Levine** [1]

ICML 2019

Introduction

- On-policy learning

Only one policy used throughout the system to both explore and select actions.
sample inefficiency

eg: policy gradient

- Off-policy learning

Two policies, one for exploring and the other for action selection.

eg: DQN

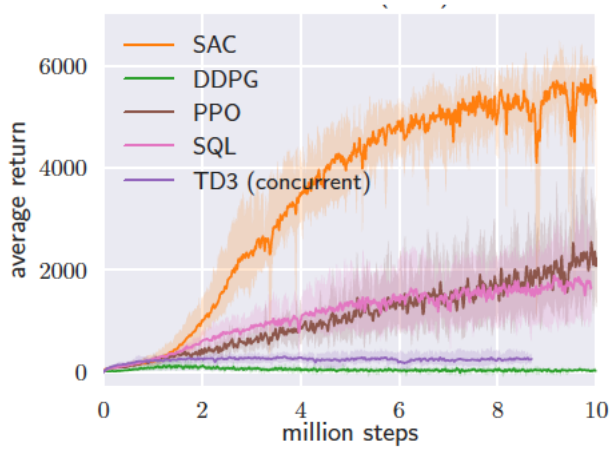
Introduction

- Meta-Reinforcement Learning

The agent can leverage varied experiences from previous tasks to adapt quickly to the new task at hand.

Motivation

- Most meta-learning RL systems use on-policy learning. The general problem with on-policy learning is sample inefficiency.

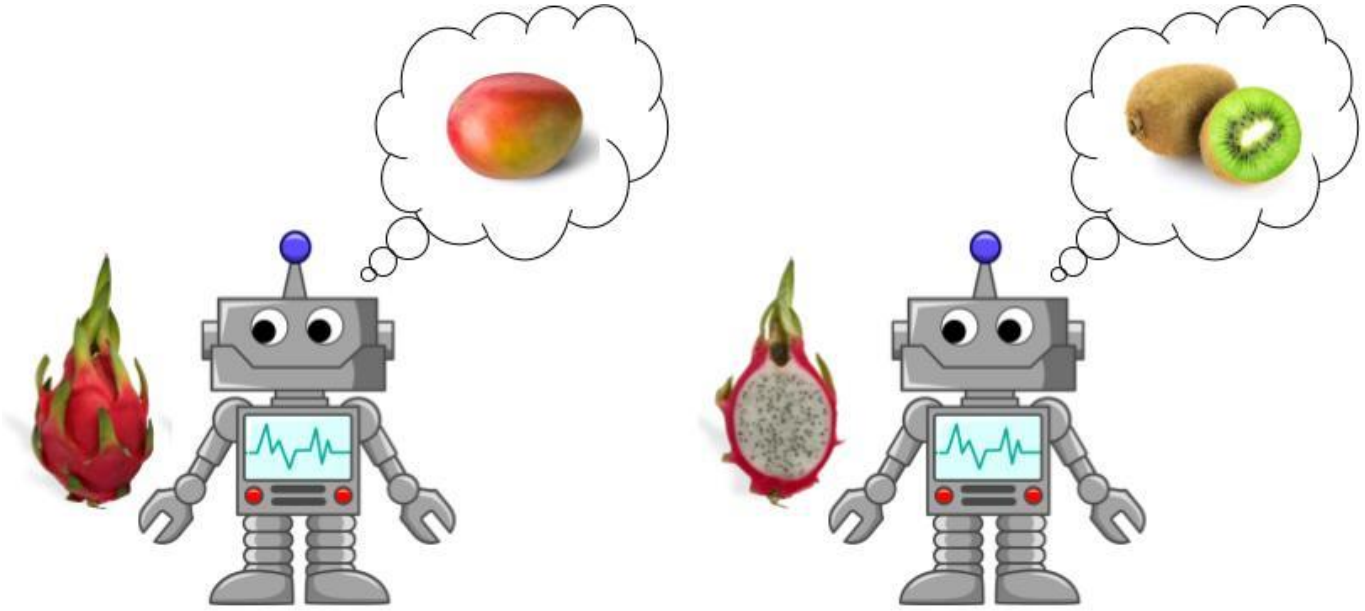


Off-policy RL (SAC) is more sample efficient than on-policy approaches (PPO) by 1-2 orders of magnitude (PPO eventually nears SAC performance), figure from Haarnoja et al. 2018.

Tackle the problem of efficient off-policy meta-reinforcement learning.

Method

- Meta-training : learn a probabilistic encoder that accumulates the necessary statistics from past experience into the context variables.
- Meta-testing : the context variables can be sampled and held constant for the duration of an episode, enabling temporally-extended exploration.



Method-Context Variable

A task : $\mathcal{T} = \{p(\mathbf{s}_0), p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t), r(\mathbf{s}_t, \mathbf{a}_t)\}$

$\mathbf{c}_n^{\mathcal{T}} = (\mathbf{s}_n, \mathbf{a}_n, r_n, \mathbf{s}'_n)$: one transition in the task $\mathcal{T}$

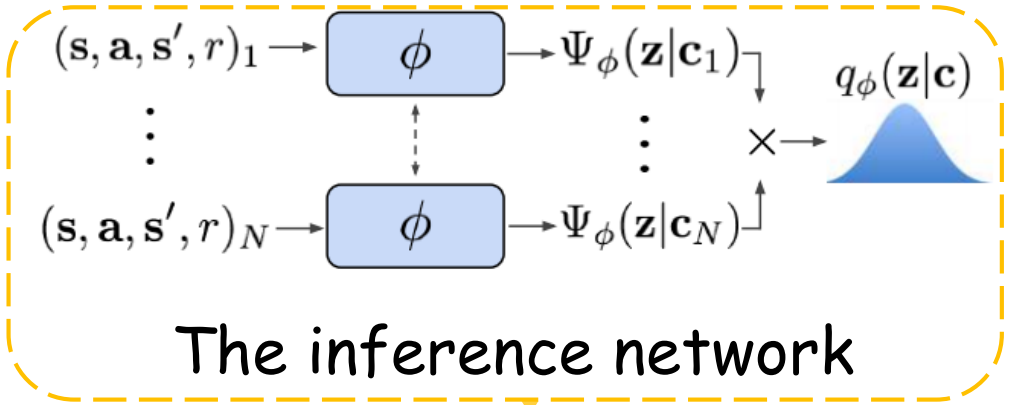
$\mathbf{z}$: latent probabilistic context variable

Gaussian factors

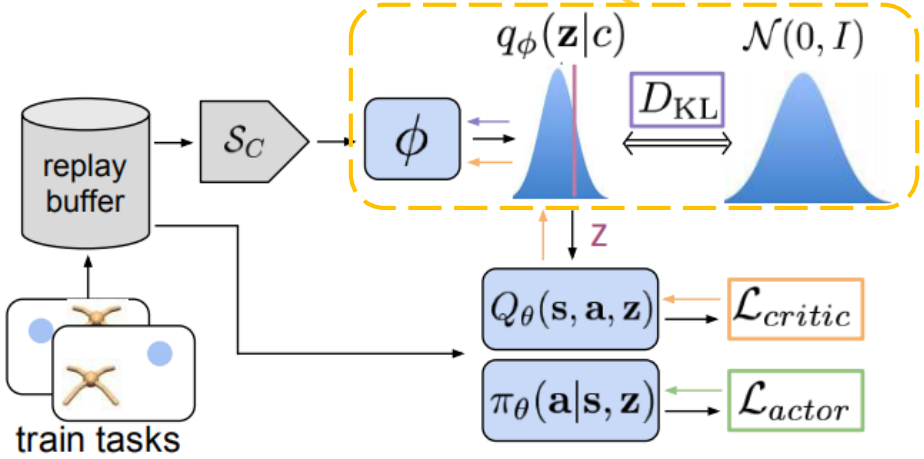
$\Psi_\phi(\mathbf{z}|\mathbf{c}_n) = \mathcal{N}(f_\phi^\mu(\mathbf{c}_n), f_\phi^\sigma(\mathbf{c}_n))$

$q_\phi(\mathbf{z}|\mathbf{c}_{1:N}) \propto \Pi_{n=1}^N \Psi_\phi(\mathbf{z}|\mathbf{c}_n)$

permutation-invariant function of prior experience



The inference network



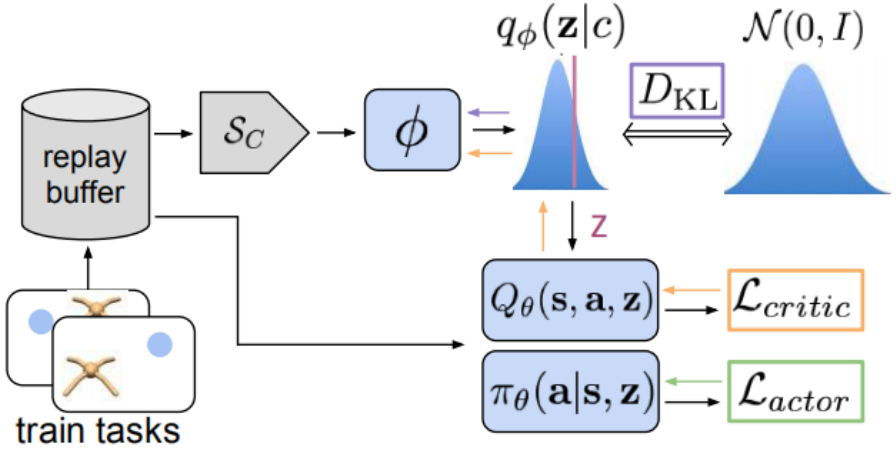
Meta-training procedure

Method-Meta-training

Algorithm 1 PEARL Meta-training

Require: Batch of training tasks $\{\mathcal{T}_i\}_{i=1...T}$ from $p(\mathcal{T})$, learning rates $\alpha_1, \alpha_2, \alpha_3$

Initialize replay buffers $\mathcal{B}^i$ for each training task
while not done **do**
  for each $\mathcal{T}_i$ **do**
    Initialize context $\mathbf{c}^i = \{\}$
    for $k = 1, \ldots, K$ **do**
      Sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$
      Gather data from $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ and add to $\mathcal{B}^i$
      Update $\mathbf{c}^i = \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)\}_{j:1...N} \sim \mathcal{B}^i$
    end for
  end for
  for step in training steps **do**
    for each $\mathcal{T}_i$ **do**
      Sample context $\mathbf{c}^i \sim \mathcal{S}_c(\mathcal{B}^i)$ and RL batch $b^i \sim \mathcal{B}^i$
      Sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$
      $\mathcal{L}^i_{actor} = \mathcal{L}_{actor}(b^i, \mathbf{z})$
      $\mathcal{L}^i_{critic} = \mathcal{L}_{critic}(b^i, \mathbf{z})$
      $\mathcal{L}^i_{KL} = \beta D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{c}^i)||r(\mathbf{z}))$
    end for
    $\phi \leftarrow \phi - \alpha_1 \nabla_\phi \sum_i \left(\mathcal{L}^i_{critic} + \mathcal{L}^i_{KL}\right)$
    $\theta_\pi \leftarrow \theta_\pi - \alpha_2 \nabla_\theta \sum_i \mathcal{L}^i_{actor}$
    $\theta_Q \leftarrow \theta_Q - \alpha_3 \nabla_\theta \sum_i \mathcal{L}^i_{critic}$
  end for
end while



Meta-training procedure

$$\mathcal{L}_{actor} = \mathbb{E}_{\substack{\mathbf{s}\sim\mathcal{B},\mathbf{a}\sim\pi_\theta \\ \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c})}}\left[D_{\mathrm{KL}}\left(\pi_\theta(\mathbf{a}|\mathbf{s}, \bar{\mathbf{z}}) \middle\| \frac{\exp(Q_\theta(\mathbf{s}, \mathbf{a}, \bar{\mathbf{z}}))}{\mathcal{Z}_\theta(\mathbf{s})}\right)\right]$$

$$\mathcal{L}_{critic} = \mathbb{E}_{\substack{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{B} \\ \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c})}}[Q_\theta(\mathbf{s}, \mathbf{a}, \mathbf{z}) - (r + \bar{V}(\mathbf{s}', \bar{\mathbf{z}}))]^2$$

Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor

Method-Meta-testing/Adaptation

Algorithm 2 PEARL Meta-testing

Require: test task $\mathcal{T} \sim p(\mathcal{T})$

1: Initialize context $\mathbf{c}^{\mathcal{T}} = \{\}$
2: **for** $k = 1, \ldots, K$ **do**
3: Sample $z \sim q_\phi(\mathbf{z}|c^{\mathcal{T}})$
4: Roll out policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ to collect data $D_k^{\mathcal{T}} = \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)\}_{j:1\ldots N}$
5: Accumulate context $\mathbf{c}^{\mathcal{T}} = \mathbf{c}^{\mathcal{T}} \cup D_k^{\mathcal{T}}$
6: **end for**

Experiments-Sample Efficiency and Performance

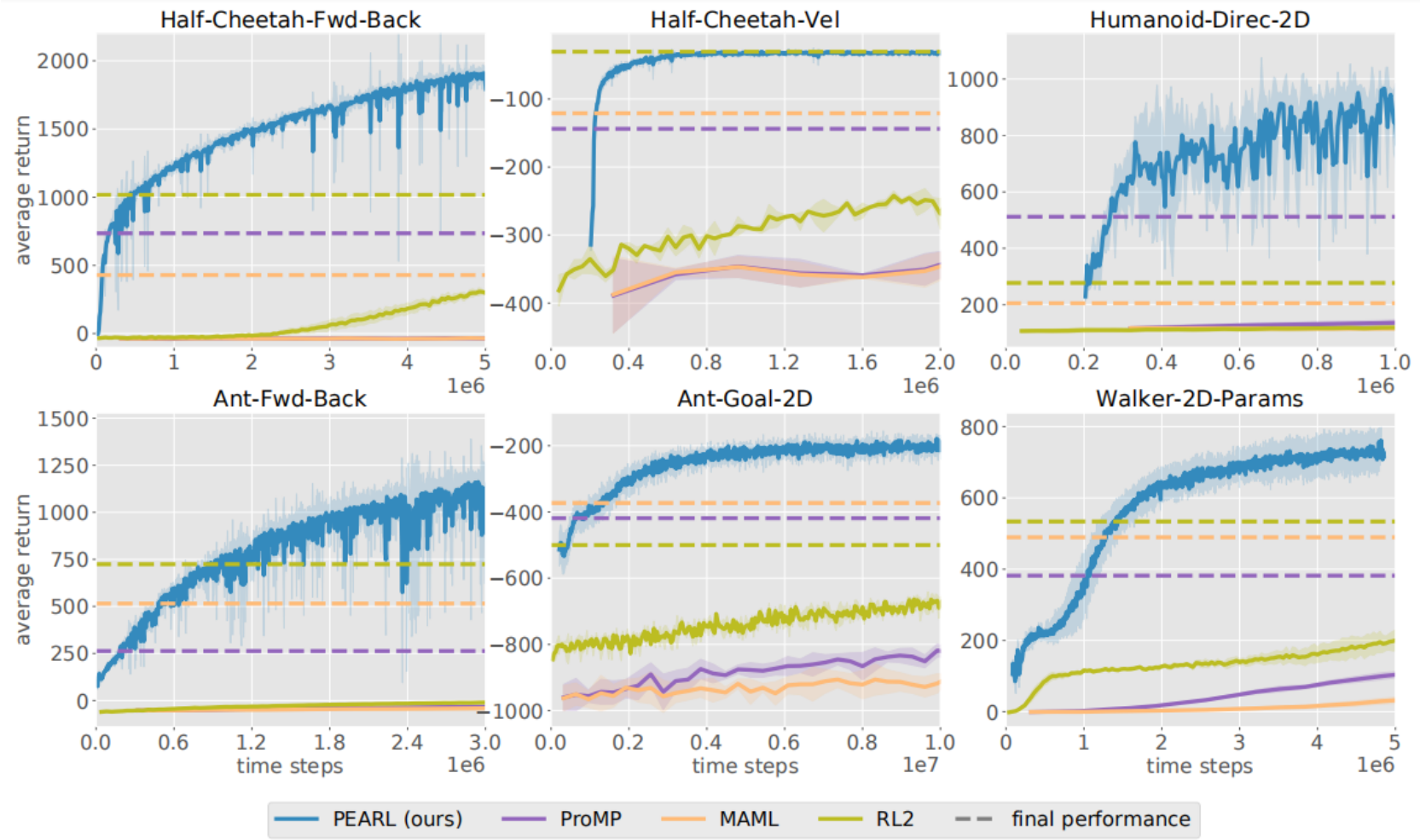
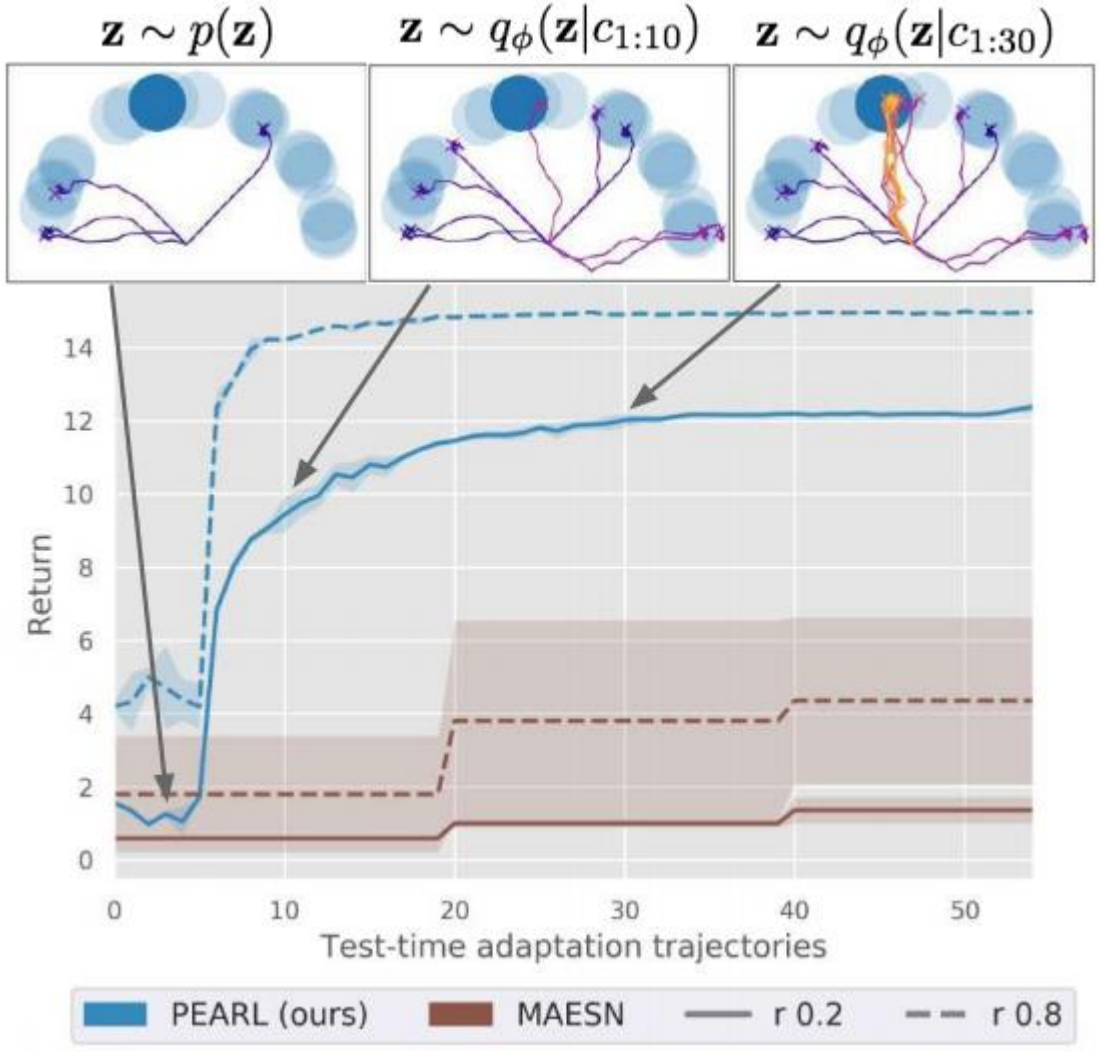


Figure 3. **Meta-learning continuous control**. Test-task performance vs. samples collected during *meta-training*. Our approach PEARL outperforms previous meta-RL methods both in terms of asymptotic performance and meta-training sample efficiency across six benchmark tasks. Dashed lines correspond to the maximum return achieved by each baseline after 1e8 steps. By leveraging off-policy data during meta-training, PEARL is 20-100x more sample efficient than the baselines, and achieves consistently better or equal final performance compared to the best performing prior method in each environment. See Appendix A for the full timescale version of this plot.

Experiments-Posterior Sampling For Exploration



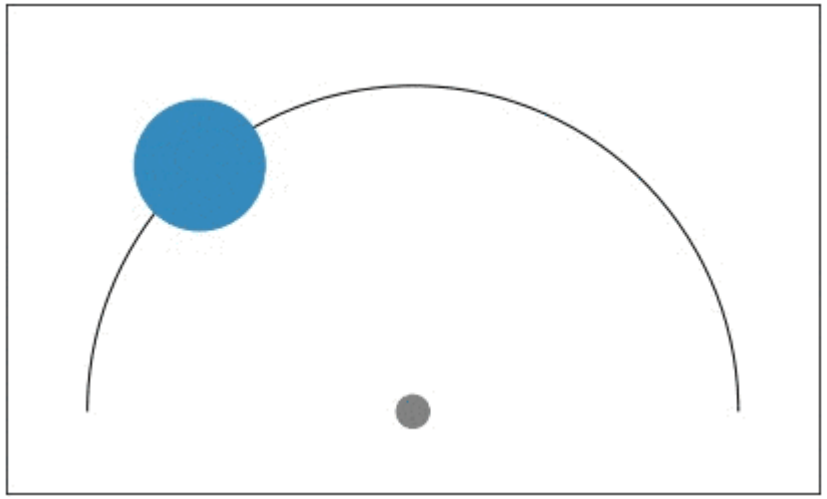


Figure 4. **Sparse 2D navigation**. The agent must navigate to a previously unseen goal (dark blue, other test goals in light blue) with reward given only when inside the goal radius – radius of 0.2 (illustrated) and 0.8 are tested here. The agent is trained to navigate to a training set of goals, then tested on a distinct set of unseen test goals. By using posterior sampling to explore efficiently, PEARL is able to start adapting to the task after collecting on average only 5 trajectories, outperforming MAESN (Gupta et al., 2018).

Experiments-Ablations

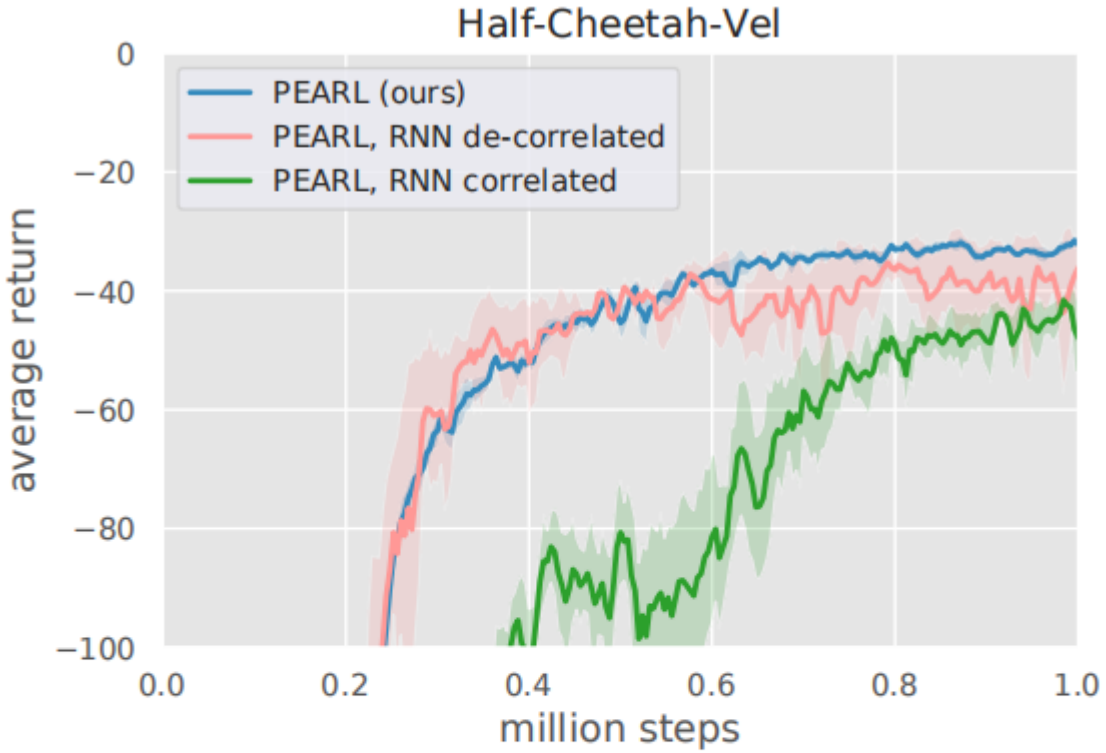


Figure 5. **Recurrent encoder ablation**. We compare our encoder architecture to a recurrent network. For the RNN, we sample context as trajectories rather than unordered transitions. Sampling the actor-critic batch as de-correlated transitions ("RNN de-correlated") fares much better than sampling trajectories ("RNN correlated"). In addition to better performance, our encoder has a computational advantage.

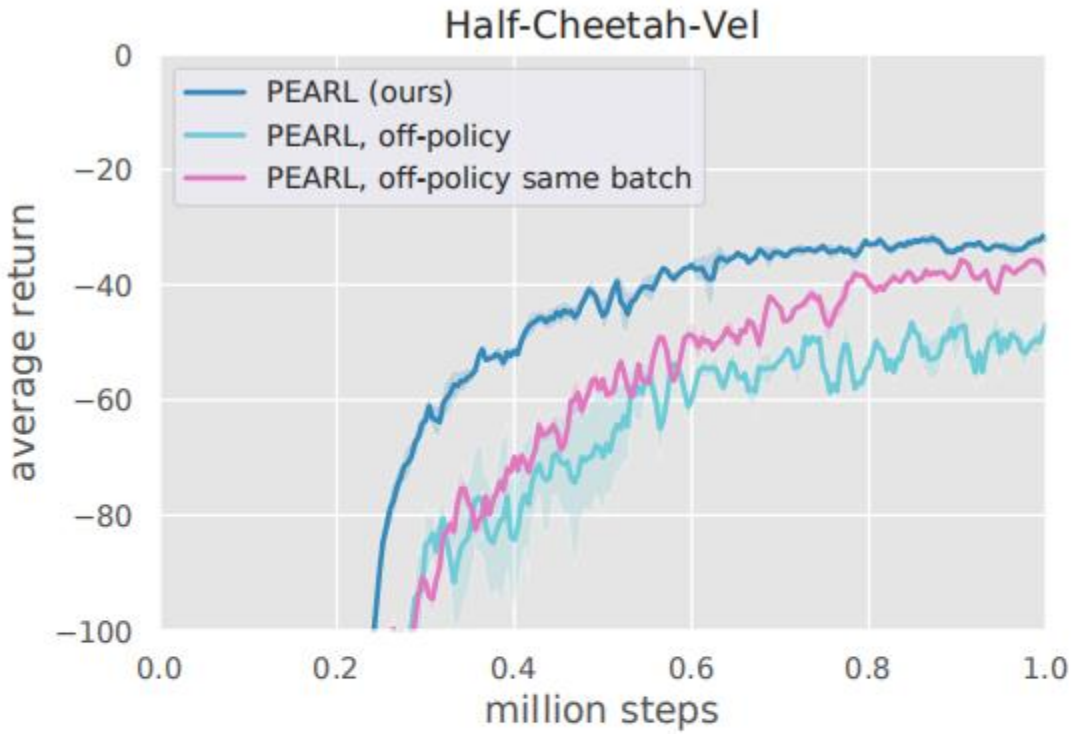


Figure 6. **Context sampling ablation**. PEARL samples context batches of recently collected transitions de-correlated with the batches sampled for training the actor-critic. We compare to sampling context from the entire history ("off-policy context"), as well as using the same sampled batch for the context and the actor-critic batch ("off-policy same batch").

Experiments-Ablations

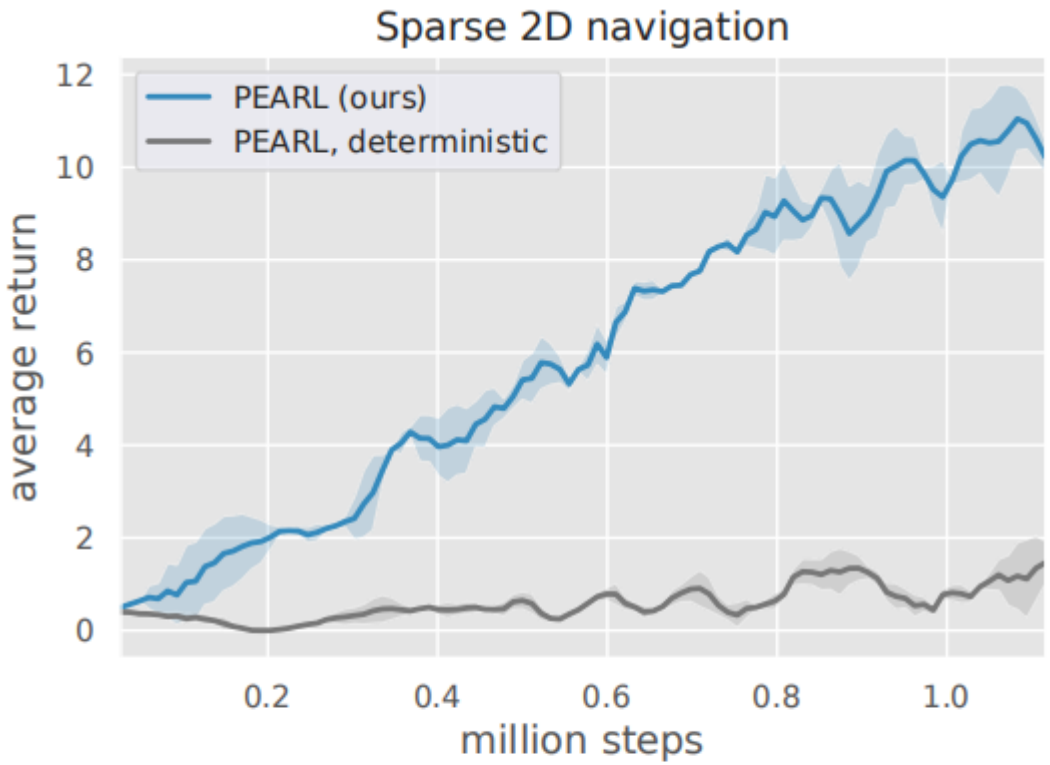


Figure 7. **Deterministic latent context**. We compare PEARL to a variant with deterministic latent context on the sparse reward 2D navigation domain. As expected, without a mechanism for reasoning about uncertainty over tasks, this approach is unable to explore effectively and performs poorly.

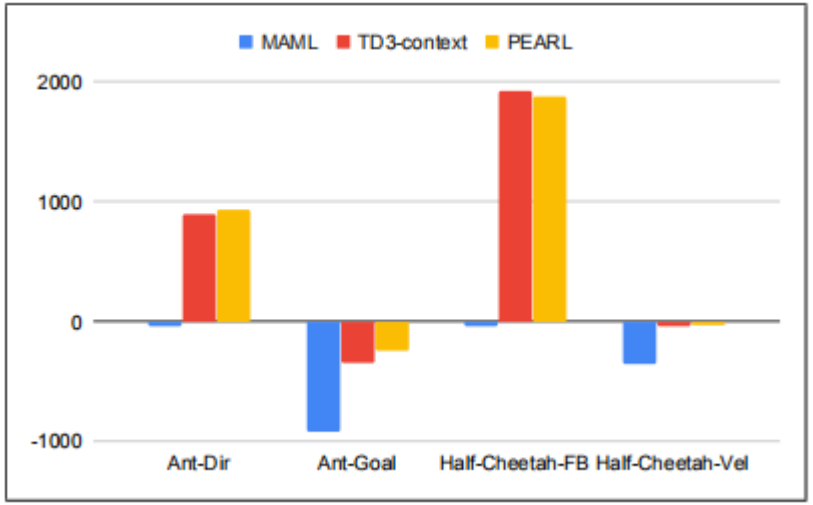
Meta-Q-Learning

Rasool Fakoor[1], **Pratik Chaudhari**[2]*, **Stefano Soatto**[1], **Alexander Smola**[1]
[1] Amazon Web Services
[2] University of Pennsylvania
Email: {fakoor, soattos, smola}@amazon.com, pratikac@seas.upenn.edu

ICLR 2020

Contribution

- Q-learning is competitive with state-of-the-art meta-RL algorithms if given access to a context variable that is a representation of the past trajectory.
- A multi-task objective to maximize the average reward across the training tasks is an effective method to meta-train RL policies.
- Past data from the meta-training replay buffer can be recycled to adapt the policy on a new task using off-policy updates.



$$\widehat{\theta}_{\text{meta}} = \arg\max_{\theta} \frac{1}{n} \sum_{k=1}^{n} \mathop{\mathbb{E}}_{\tau \sim D^k} \left[\ell^k(\theta) \right]$$

Method-Context Variable

- recurrent context variable z_t

Set z_t to the hidden state at time t of a Gated Recurrent Unit (GRU) model.

z_t depends on $\{(x_i, u_i, r_i)\}_{i<t}$.

policies	$u_\theta(x)$	→	$u_\theta(x, z)$
value functions	$q_\varphi(x, u)$	→	$q_\varphi(x, u, z)$

Method-Meta-training

Algorithm 1: MQL - Meta-training

Input: Set of training tasks $\mathcal{D}_{\text{meta}}$

Initialize the replay buffer
Initialize parameters θ of an off-policy method, e.g., TD3
while *not done* **do**
    // Rollout and update policy
    Sample a task $D \sim \mathcal{D}_{\text{meta}}$
    Gather data from task D using policy π_θ while feeding transitions through context GRU. Add trajectory to the replay buffer.
    $\mathcal{b} \leftarrow$ Sample mini-batch from buffer
    Update parameters θ using mini-batch $\mathcal{b}$ and Eqn. (15)
$\theta_{\text{meta}} \leftarrow \theta$
return θ_{meta} , replay buffer

$$\widehat{\theta}_{\text{meta}} = \arg\min_{\theta} \frac{1}{n} \sum_{k=1}^{n} \mathop{\mathbb{E}}_{\tau \sim D^k} \left[\text{TD}^2(\theta) \right];$$

Method-Meta-training

Objective

$$\widehat{\theta}_{\text{meta}} = \arg\max_{\theta} \frac{1}{n} \sum_{k=1}^{n} \mathop{\mathbb{E}}_{\tau \sim D^k} \left[\ell^k(\theta) \right]$$

TD3

$$\widehat{\theta}_{\text{meta}} = \arg\min_{\theta} \frac{1}{n} \sum_{k=1}^{n} \mathop{\mathbb{E}}_{\tau \sim D^k} \left[\text{TD}^2(\theta) \right];$$

$$\text{TD}^2(\theta) = \left(q_\varphi(x, u) - r^k - \gamma \, q_\varphi(x', u_\theta(x')) \right)^2$$

Analysis

MAML : $\ell^k_{\text{meta}}(\theta) = \ell^k(\theta + \alpha \nabla_\theta \ell^k(\theta))$

Taylor series expansion & Gradient:

$$\nabla \ell^k_{\text{meta}}(\theta) = \nabla \ell^k(\theta) + 2\alpha(m-1) \left(\nabla^2 \ell^k(\theta) \right) \nabla \ell^k(\theta) + \mathcal{O}(\alpha^2).$$

gradient

$$\ell^k(\theta) + \alpha(m-1) \| \nabla \ell^k(\theta) \|_2^2$$

Method-Meta-testing/Adaptation

- Update the policy using the new data

$$\arg\max_{\theta} \left\{ \mathop{\mathbb{E}}_{\tau \sim D^{\text{new}}} \left[\ell^{\text{new}}(\theta) \right] - \frac{\lambda}{2} \left\| \theta - \widehat{\theta}_{\text{meta}} \right\|_2^2 \right\}. \tag{18}$$

- Exploits the meta-training replay buffer

$$\arg\max_{\theta} \left\{ \mathop{\mathbb{E}}_{\tau \sim \mathcal{D}_{\text{meta}}} \left[\beta(\tau; D^{\text{new}}, \mathcal{D}_{\text{meta}})\, \ell^{\text{new}}(\theta) \right] - \frac{\lambda}{2} \left\| \theta - \widehat{\theta}_{\text{meta}} \right\|_2^2 \right\}. \tag{19}$$

$$\beta(\tau; D^{\text{new}}, \mathcal{D}_{\text{meta}}) = \frac{P(\tau \in D^{new})}{P(\tau \in D_{meta})}$$

- Effective Sample Size

Normalized Effective Sample Size ($\widehat{\text{ESS}}$): A related quantity to $\beta(x)$ is the normalized Effective Sample Size ($\widehat{\text{ESS}}$) which we define as the relative number of samples from the target distribution $p(x)$ required to obtain an estimator with performance (say, variance) equal to that of the importance sampling estimator (10). It is not possible to compute the $\widehat{\text{ESS}}$ without knowing both densities

$$\widehat{\text{ESS}} = \frac{1}{m} \frac{\left(\sum_{k=1}^{m} \beta(x_k)\right)^2}{\sum_{k=1}^{m} \beta(x_k)^2} \in [0, 1]$$

$$\lambda = 1 - \widehat{\text{ESS}}$$

$$\arg\max_{\theta} \left\{ \mathop{\mathbb{E}}_{\tau \sim D^{\text{new}}} \left[\ell^{\text{new}}(\theta) \right] + \mathop{\mathbb{E}}_{\tau \sim \mathcal{D}_{\text{meta}}} \left[\beta(\tau; D^{\text{new}}, \mathcal{D}_{\text{meta}})\, \ell^{\text{new}}(\theta) \right] - \left(1 - \widehat{\text{ESS}}\right) \left\| \theta - \widehat{\theta}_{\text{meta}} \right\|_2^2 \right\}$$

Method-Meta-testing/Adaptation

Algorithm 2: MQL - Adaptation

Input: Test task D, meta-training replay buffer, meta-trained policy θ_{meta}

1 Initialize temporary buffer buf
2 $\theta \leftarrow \theta_{meta}$
3 buf $\leftarrow$ Gather data from D using $\pi_{\theta_{meta}}$
4 Update Eqn. (18) using buf
5 Fit $\beta(D)$ using buf and meta-training replay buffer using Eqn. (12)
6 Estimate $\widehat{\text{ESS}}$ using $\beta(D)$ using Eqn. (13)
7 **for** $i \leq n$ **do**
8 $\quad$ $\mathcal{b} \leftarrow$ sample mini-batch from meta-training replay buffer
9 $\quad$ Calculate β for $\mathcal{b}$
10 $\quad$ Update θ using Eqn. (19)
11 Evaluate θ on a new rollout from task D
12 **return** θ

$$\arg\max_{\theta} \left\{ \mathop{\mathbb{E}}_{\tau \sim D^{\text{new}}} \left[\ell^{\text{new}}(\theta) \right] - \frac{\lambda}{2} \|\theta - \widehat{\theta}_{\text{meta}}\|_2^2 \right\}. \tag{18}$$

$$\beta(\tau; D^{\text{new}}, \mathcal{D}_{\text{meta}}) = \frac{P(\tau \in D^{new})}{P(\tau \in D_{meta})}$$

$$\widehat{\text{ESS}} = \frac{1}{m} \frac{\left(\sum_{k=1}^{m} \beta(x_k)\right)^2}{\sum_{k=1}^{m} \beta(x_k)^2} \in [0, 1] \tag{13}$$

$$\arg\max_{\theta} \left\{ \mathop{\mathbb{E}}_{\tau \sim \mathcal{D}_{\text{meta}}} \left[\beta(\tau; D^{\text{new}}, \mathcal{D}_{\text{meta}}) \, \ell^{\text{new}}(\theta) \right] - \frac{\lambda}{2} \|\theta - \widehat{\theta}_{\text{meta}}\|_2^2 \right\}. \tag{19}$$

Experiments

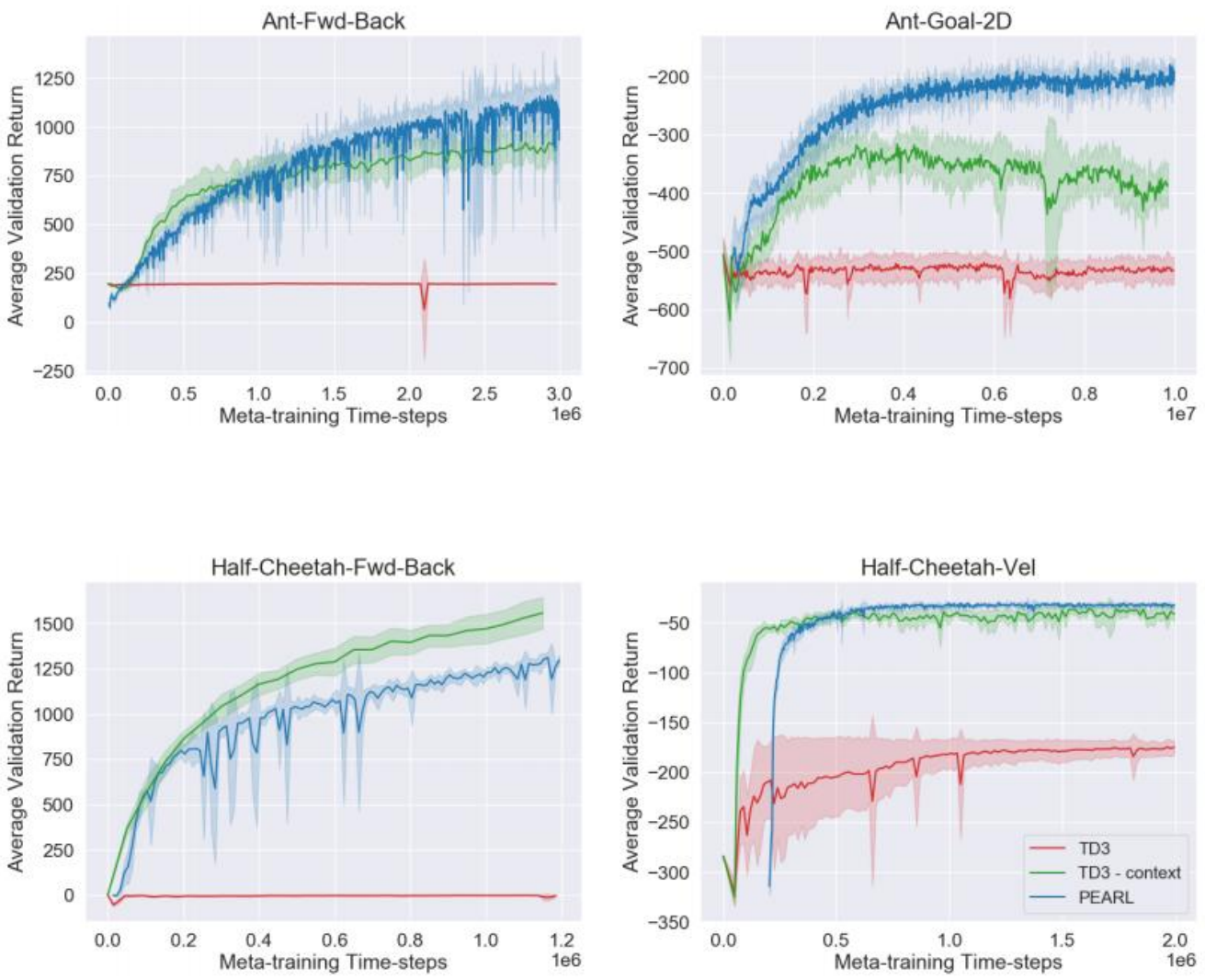


Figure 2: **Average undiscounted return of TD3 and TD3-context compared with PEARL for validation tasks from four meta-RL environments.** The agent fails to learn if the policy is conditioned only on the state. In contrast, everything else remaining same, if TD3 is provided access to context, the rewards are much higher. In spite of not adaptating on the validation tasks, TD3-context is comparable to PEARL.

Experiments

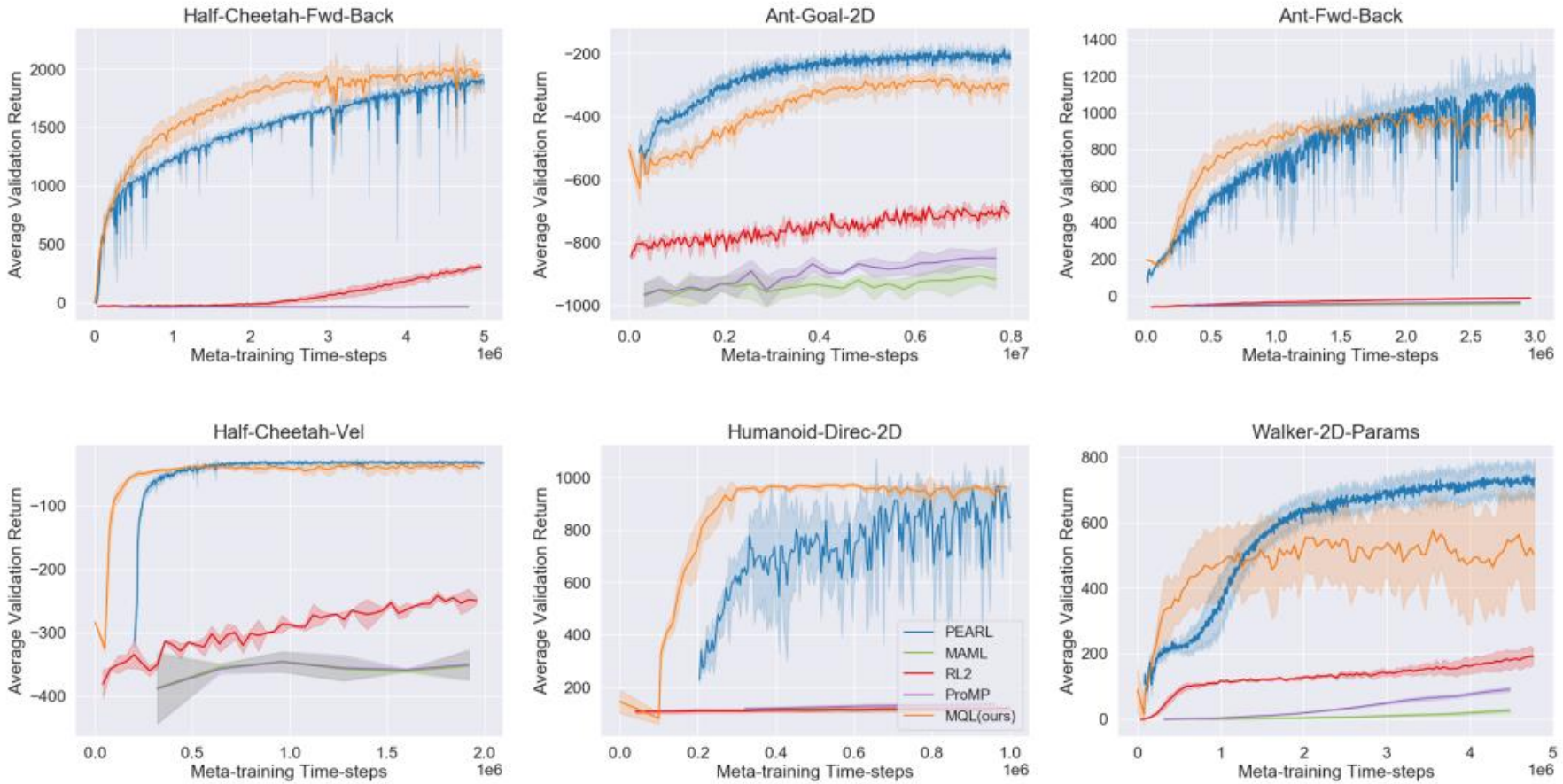
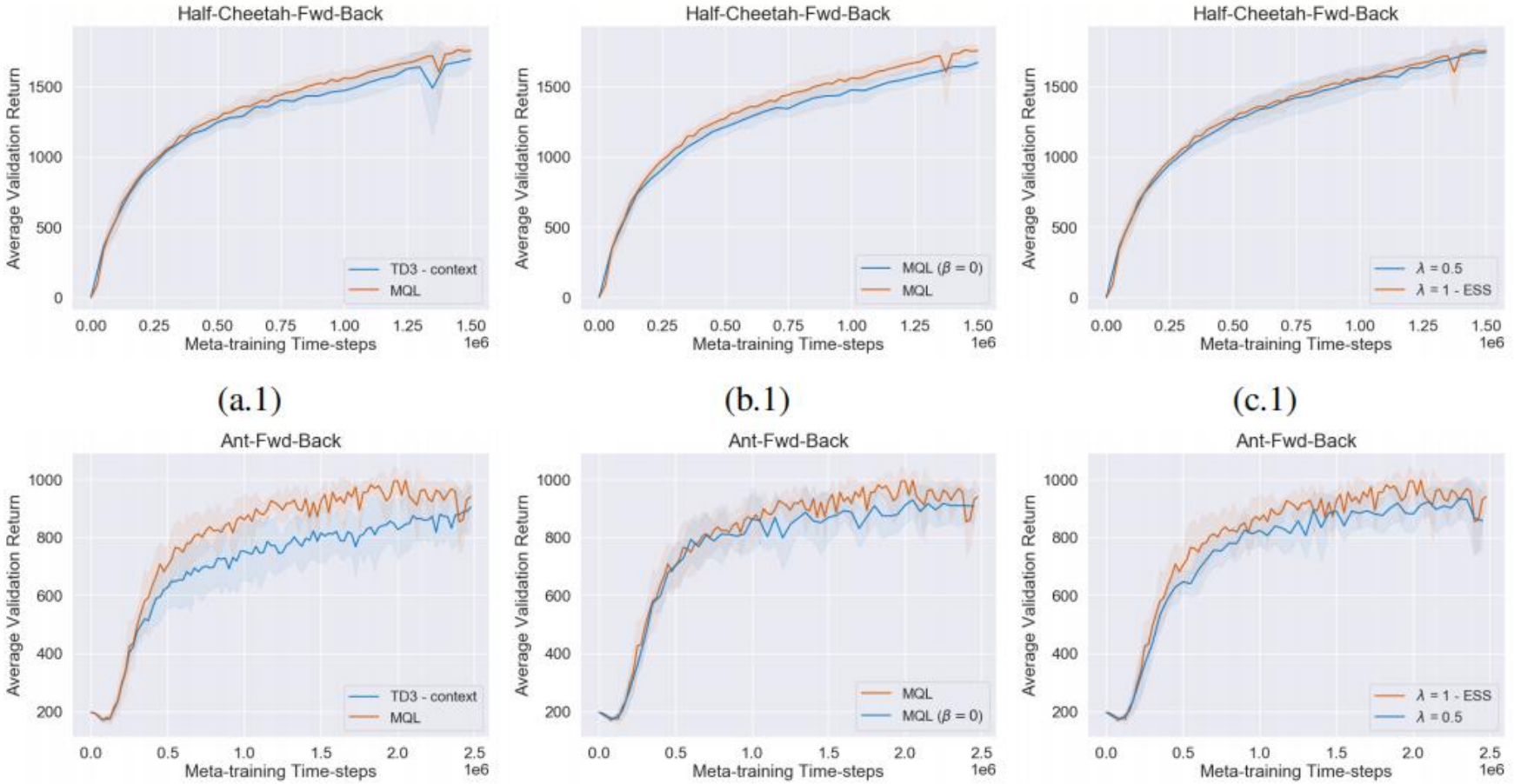


Figure 3: **Comparison of the average undiscounted return of MQL (orange) against existing meta-RL algorithms on continuous-control environments.** We compare against four existing algorithms, namely MAML (green), RL2 (red), PROMP (purple) and PEARL (blue). In all environments except Walker-2D-Params and Ant-Goal-2D, MQL is better or comparable to existing algorithms in terms of both sample complexity and final returns.

Experiments-Ablations



(a) MQL vs. TD3-context (b) MQL vs. MQL ($\beta = 0$) (c) Choice of $\lambda = 1 - \widehat{\text{ESS}}$

Figure 4: **Ablation studies to examine various components of MQL.**

Thanks