Efficient Off-Policy Meta RL via Probabilistic Context:

`https://arxiv.org/pdf/1903.08254.pdf`

Paper Summary Notes

1 Main Ideas, High Level Assumptions

Central Question of the Paper: *How we can improve meta-training efficiency in meta-reinforcement learning, to minimize the computational burden of meta-training phase, while maintaining good few-shot adaptation to new tasks at meta-test time?*

1.1 Problem Statement

- typically, meta learning procedures focus on quick adaptation, and ignore the computational cost of good meta-training
- recall that meta-training involves interacting with a *distribution* of tasks $M \sim p(\mathcal{M})$, aggregating information $\mathcal{I}_M$ based on these interactions, then leveraging this for quick adaptation to a new task $M_{test} \sim p(\mathcal{M})$ via conditioned policy $\pi(a|s, \mathcal{I})$

2 Approach

- the paper takes a probabilistic approach, where $\mathcal{I}_M \equiv \mathbf{c} = (s_i, a_i, s_{i+1}, r_i)_{i=1,\dots,N}$ the *context* is given sampled trajectories for interacting with an MDP M, and we encode the relevant information in *latent variables* $\mathcal{Z}$
- During meta-training we learn a probabilistic encoder that estimates the posterior over the latent variables
- During meta-test context variables can be sampled and held constant for tempoerally extended exploration
- *sample task hypothesis, attempt the task, re-evalaute task hypothesis*

2.1 Probabilistic Context

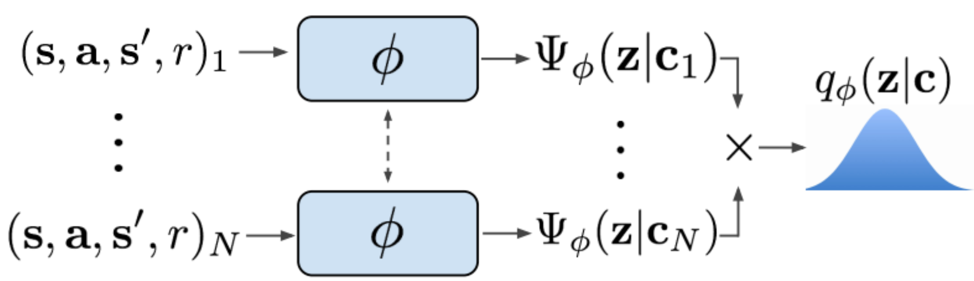


Figure 1. **Inference network architecture.** The amortized inference network predicts the posterior over the latent context variables $q_\phi(\mathbf{z}|\mathbf{c})$ as a permutation-invariant function of prior experience.

- Inference network $q_\phi(z|c)$ estimates the posterior $p(z|c)$
- The latent variables z capture knowledge about how the current task should be performed; we condition our policy as $\pi_\theta(a|s, z)$
- By markov property, the *ordering* of context transition doesn't matter, so we assume independence over trajectory:

$$q_\phi(z|c_{1:N}) \propto \prod_{n=1}^{N} \Psi_\phi(z|c_n)$$

- For tractability, assume gassian factors:

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

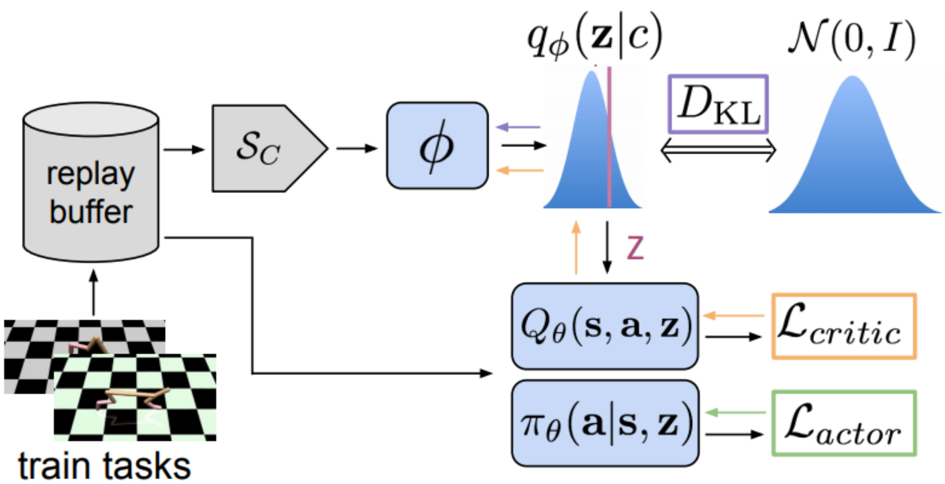


Figure 2. **Meta-training procedure.** The inference network q_ϕ uses context data to infer the posterior over the latent context variable Z, which conditions the actor and critic, and is optimized with gradients from the critic as well as from an information bottleneck on Z. De-coupling the data sampling strategies for context ($\mathcal{S}_C$) and RL batches is important for off-policy learning.

- q is optimized under the variational lower bound:

$$\mathbb{E}_\mathcal{M}[\mathbb{E}_{z\sim q_\phi(z|c_\mathcal{M})}[\mathcal{R}(\mathcal{M}, z) + \beta D_{KL}(q_\phi(z, c_\mathcal{M})|p(z))]]$$

  - $\mathcal{R}(\mathcal{M}, z)$ can be the liklihood of some MDP specific quantity (ex. value function, state liklihood, reward liklihood, actor return) [1]
  - second term ensures structure in latent space by penalizing deviance from gaussian factors

2.2 Exploration

- acting optimally with regards to a random MDP allows for deep exploration
- at test time we sample z from prior and execute for an episode, then we update posterior belief over mdp using experience tuples
- *act more and more optimally as our belief narrows*

[1]they found the best was backpropogating through the critic TD-residual

2.3 Off-Policy RL

- insight: data used to train encoder need not be the same data used to train rl agent
- so we train actor,critic using off-policy data from entire replay buffer $\mathcal{B}$
- define sampler $\mathcal{S}_c$ which takes most recent experience to train encoder
- in this way we get efficiency in meta-training since we have sample reuse for rl agent, while maintaining online belief propogation on current task for encoder

2.4 Pseudo Code

Algorithm 1 PEARL Meta-training

Require: Batch of training tasks $\{\mathcal{T}_i\}_{i=1\ldots T}$ 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 $\mathbf{c}^i = \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)\}_{j:1\ldots N} \sim \mathcal{B}^i$
    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

Algorithm 2 **PEARL Meta-testing**

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

Initialize context $\mathbf{c}^{\mathcal{T}} = \{\}$

for $k = 1, \dots, K$ **do**

  Sample $z \sim q_\phi(\mathbf{z}|c^{\mathcal{T}})$

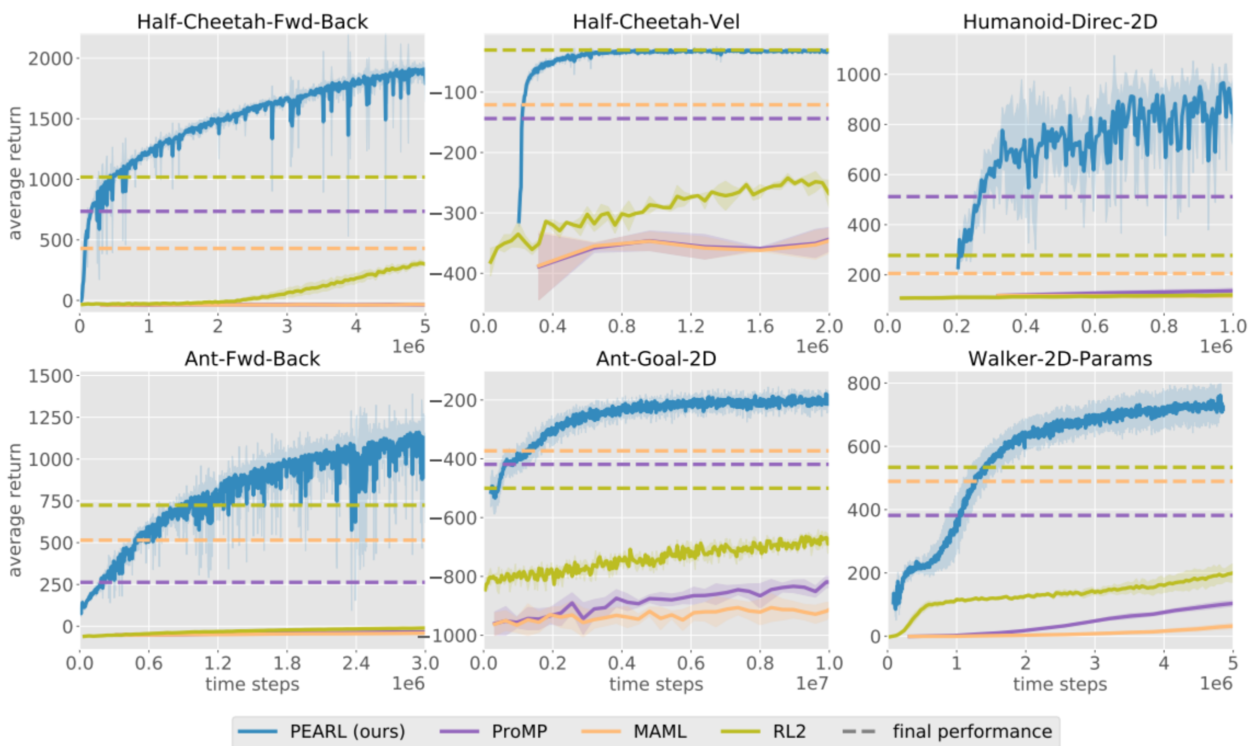
  Roll out policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ to collect data $D_k^{\mathcal{T}} = \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)\}_{j:1\dots N}$

  Accumulate context $\mathbf{c}^{\mathcal{T}} = \mathbf{c}^{\mathcal{T}} \cup D_k^{\mathcal{T}}$

end for

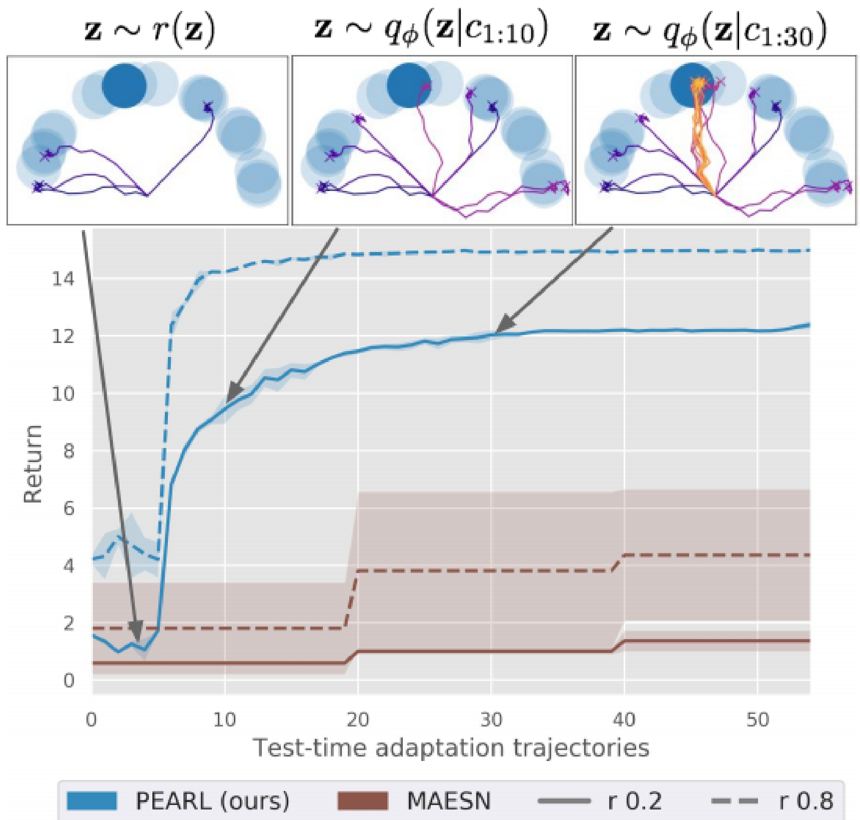
3 Results

3.1 Sample Efficiency and Performance



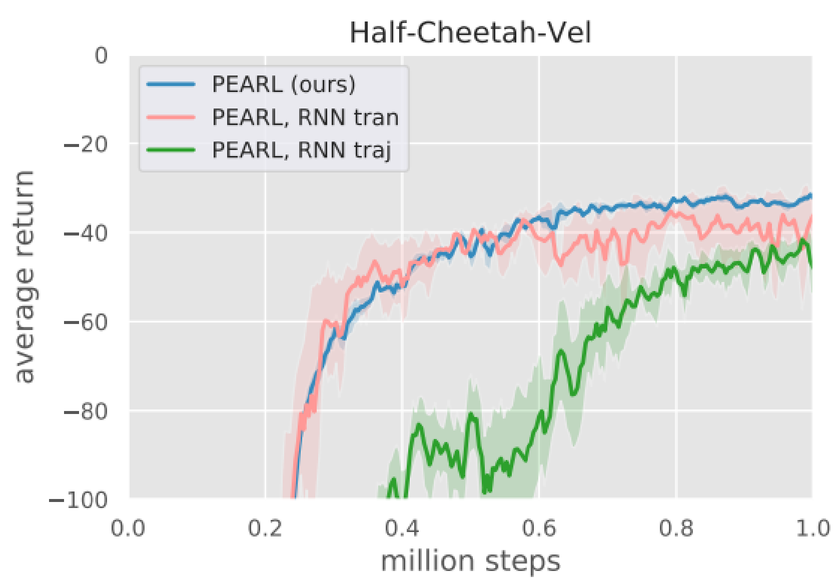
- PEARL is much more sample efficient then recurrent based and gradient based baselines, and achieves as good or better asymptotic performance

3.2 Exploration



- structured exploration from latent variable contexts allows for quick adaptation at test time

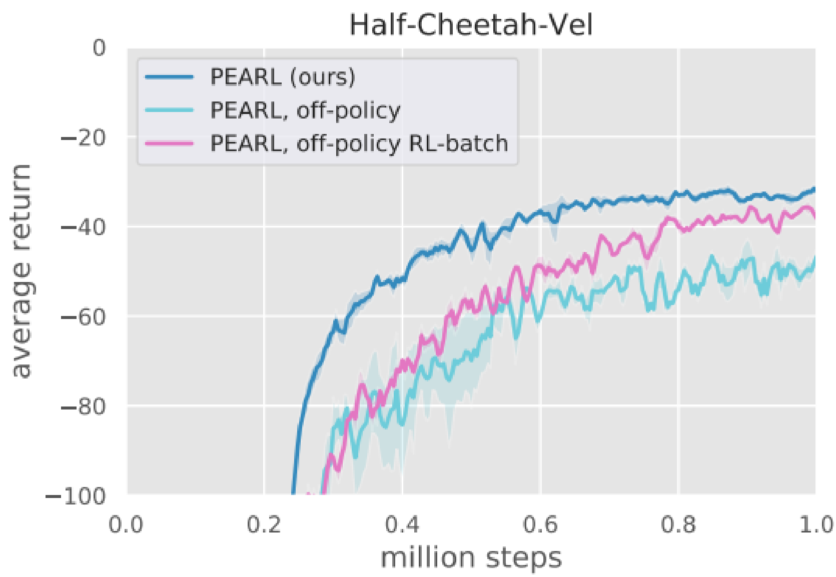
3.3 Ablation



3.3.1 Inference Network Architecture

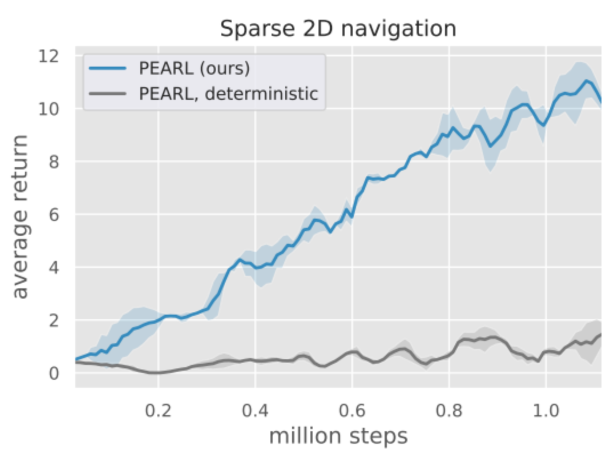
- using recurrent architecture over trajectories does much worse, which highlights the need to decouple RL control, and inference over context
- using recurrent architecture over transitions does just as good, but is much more costly

3.3.2 Context Sampling



- sampling contexts for training the probabilistic encoder does worse if we sample from entire replay buffer (off-policy) and when using same data as RL agent (off-policy RL batch) which highlights the need to decouple RL control, and inference over context

3.3.3 Deterministic Context



- using point estimates, we remove the exploration over MDP's, and we fail on sparse rewards tasks