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

Rakelly, K., Zhou, A., Quillen, D., Finn, C., & Levine, S

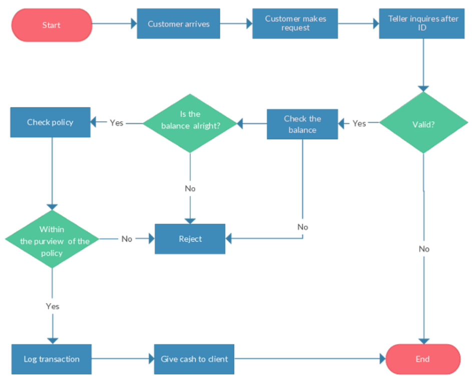
ICML, 2019

Presented by: Egill Ian Gudmundsson

UNIVERSITY OF WATERLOO

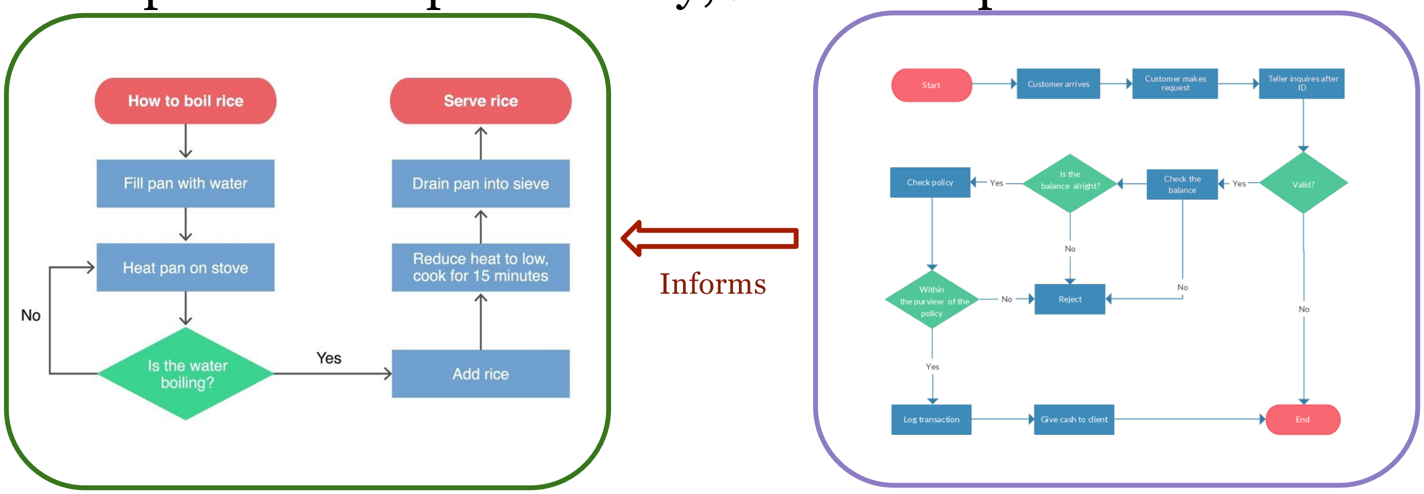
Some Terminology

- **On-policy learning:** Only one policy used throughout the system to both explore and select actions. Not optimal because policy covers exploration as well, but less costly.



Some Terminology

- **Off-policy learning:** Two policies, one for exploring and the other for action selection. Expensive computationally, but more optimal solution achieved with fev



Target Policy
(Exploitation)

Behaviour Policy
(Exploration)

Some Terminology

- **Meta-Reinforcement Learning:** First train a reinforcement learning system to do a task, then train it to do a second different task
- The hope is that some of its ability to do the first will help it learn how to do the second
- I.e. we will converge faster on a solution for the second using knowledge from the first
- If this happens, it is called meta-learning. Learning how to learn.
- Depending on the system, pre-training can be meta-learning

Problem Definition

- Most meta-learning RL systems use on-policy learning
- The general problem with on-policy learning is sample inefficiency
- There is **meta-training efficiency** for other tasks and **adaptation efficiency** for the task at hand
- Ideally, both should be good. That is, we want few-shot learning.
- Current methods would use off-policy during training and then on-policy during inference. But this might lead to overfitting in off-policy methods (different real data).
- How can current solutions be improved? The authors propose Probabilistic Embeddings for Actor-critic RL (**PEARL**)

PEARL Method

- We have a set of tasks T, each of which consists of an initial state distribution, initial transition distribution and initial reward function
- Each sample is a tuple referred to as a context $\boldsymbol{c} = (\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}')$ and each task has a set of size N these samples $\mathbf{c}_{1:N}$
- Now for the innovative bit: A latent (hidden) probabilistic context variable Z is added to the mix and the policy is conditioned with this variable as $\pi_\theta(\mathbf{a} \mid \mathbf{s}, \mathbf{z})$ while learning a task
- A soft actor-critic (SAC) method is used in addition to Z

The Z Variable

- How do we ensure that Z captures meta-learning properties and not other dependencies?
- An inference network $q(\mathbf{z} \mid \mathbf{c})$ is trained during the meta-training phase to estimate $p(\mathbf{z} \mid \mathbf{c})$. To sidestep the intractability, the lower bound is used for optimization $\mathbb{E}_{\mathcal{T}}[\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c}^{\mathcal{T}})}[\underbrace{\boxed{R(\mathcal{T},\mathbf{z})}}_{\text{Reward from task objective}} + \beta \underbrace{\boxed{D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{c}^{\mathcal{T}})||p(\mathbf{z}))}}_{\text{Informational bottleneck}}]]$
- Optimization is now model-free using evidence lower bound (ELBO)

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

- Use Gaussian factors to lessen impact of context size and order (permutation invariant)

The Inherent Stochasticity of Z

- The variable Z can be said to learn the uncertainty of the tasks that it is presented with, a bit similar to the beta functions in Thompson sampling
- Due to the policy relying on $\mathbf{z}$ to reach a decision, there is a degree of uncertainty that becomes less and less as the model learns more
- This initial uncertainty seems to be enough to get the model to explore in the new task, but not so much to prevent optimal convergence

Soft Actor-Critic Part

- The optimal off-policy model for this method was found to be SAC with the following loss functions:

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

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

Pseudocode

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
    $\phi \leftarrow \phi - \alpha_1 \nabla_\phi \sum_i (\mathcal{L}^i_{critic} + \mathcal{L}^i_{KL})$
    $\theta_\pi \leftarrow \theta_\pi - \alpha_2 \nabla_\theta \sum_i \mathcal{L}^i_{actor}$
    $\theta_Q \leftarrow \theta_Q - \alpha_3 \nabla_\theta \sum_i \mathcal{L}^i_{critic}$
  end for
end while

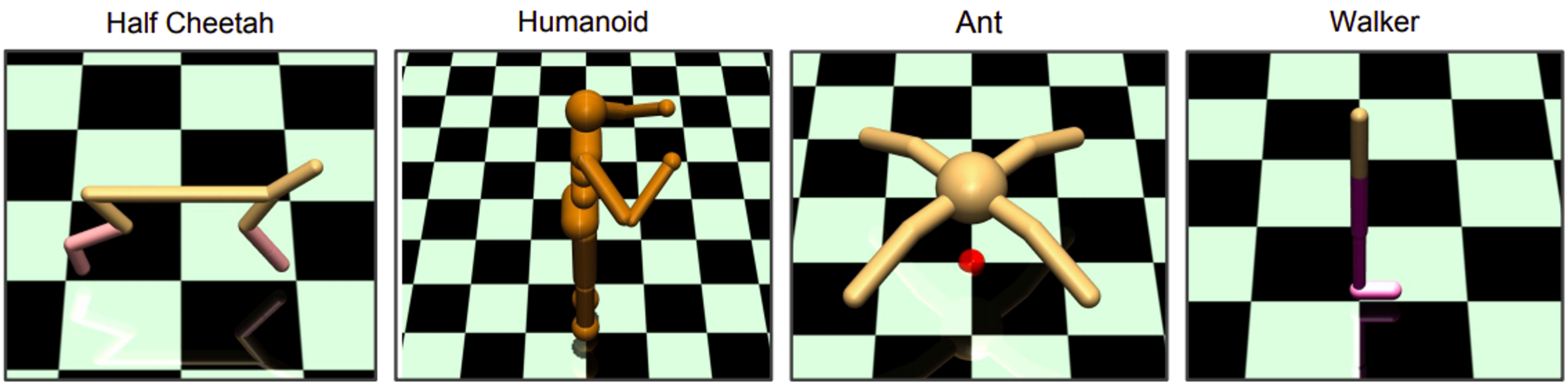
Fill our buffers with relevant data for the task

Sample using actor-critic and utilize $\mathbf{z}$

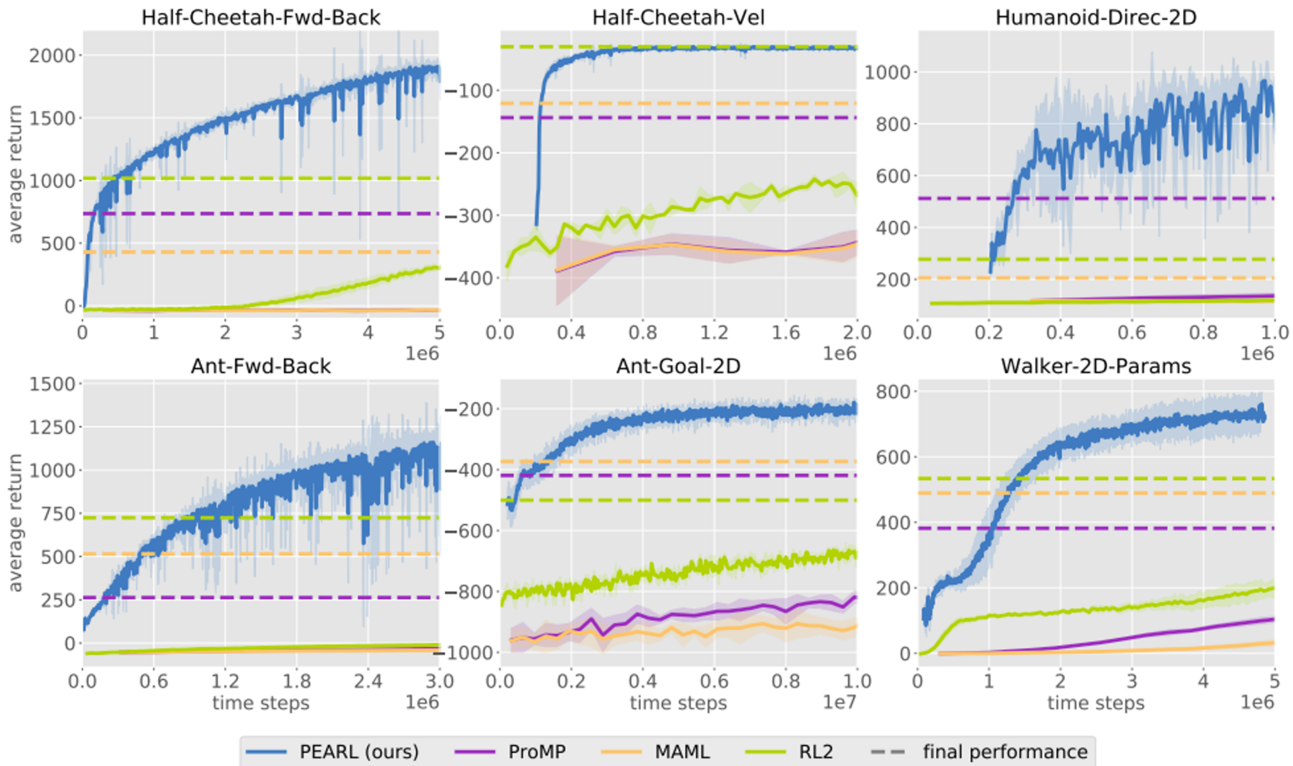
Update weights

Tasks

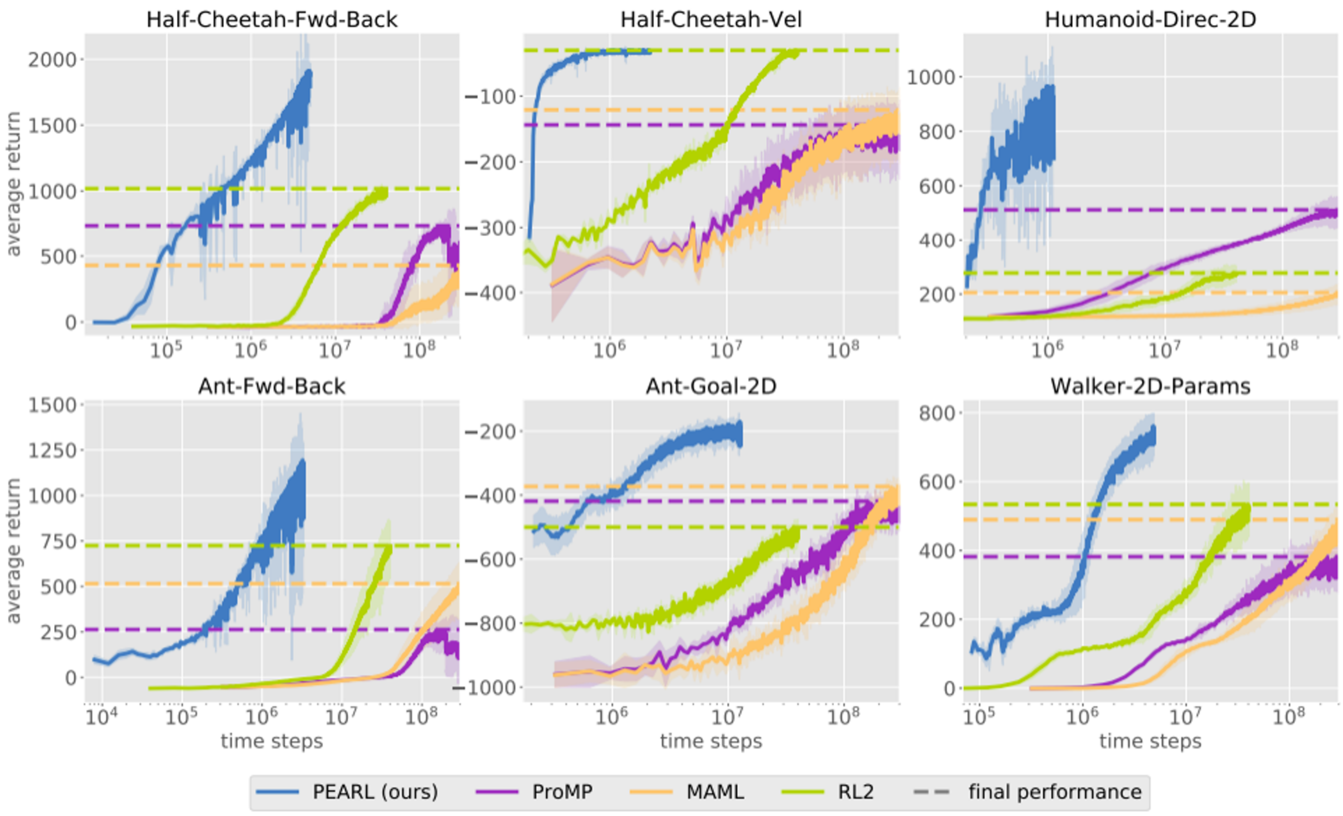
- The classic MuJoCo environment and tasks used



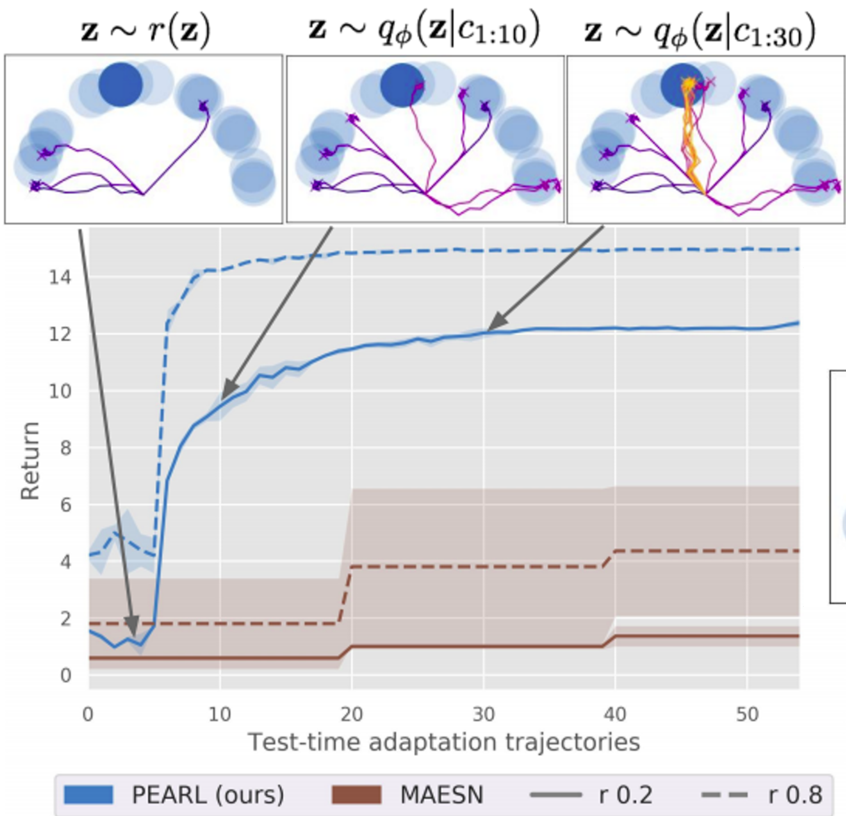
Meta-Training Results

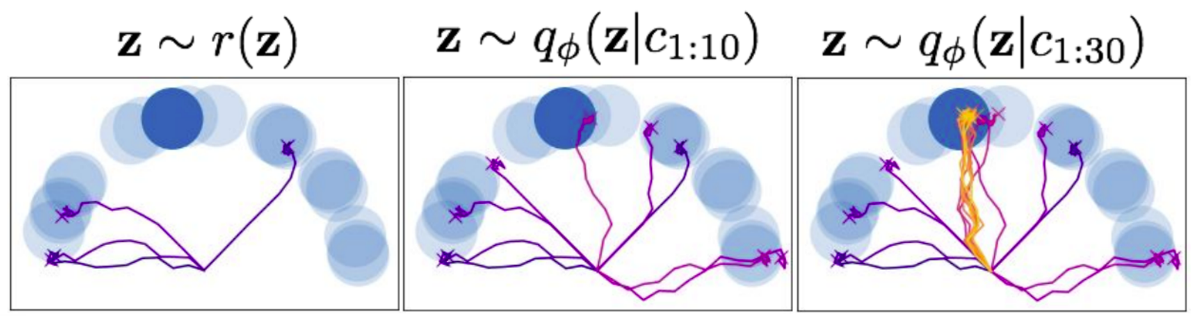


Meta-Training Results, Further Time Steps



Adaptation Efficiency Example





Anything Missing?

- Drastically improves meta-learning capabilities
- Shows that off-policy methods are usable in these circumstances
- Ablation shows that the benefits are thanks to the changes suggested
- All of these tasks are fairly similar. What about meta-training on a disparate set of tasks? Is there still an advantage?
- Most of the results are about the meta-learning step. What about adaptation efficiency in general?

References

- Rakelly et al., Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables, https://arxiv.org/pdf/1903.08254.pdf
- Vinyals et al., Matching Networks for One Shot Learning, https://arxiv.org/pdf/1606.04080.pdf
- Kingma & Welling, Auto-Encoding Variational Bayes, https://arxiv.org/pdf/1312.6114.pdf
- Haarnoja et al., Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, https://arxiv.org/pdf/1801.01290.pdf