Contrastive Learning for Context-Based Off-Policy Actor-Critic Reinforcement Learning

Bernie Wang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-219
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-219.html

December 18, 2020

Acknowledgement

I would like to thank Professor Kurt Keutzer for being my research advisor and for the opportunity to work in Adept Lab. He has provided valuable insights into my research projects and guidance throughout my time in his group. I would also like to thank Professor Joseph Gonzalez for being the second reader and providing valuable feedback on my thesis. I especially would like to thank Doctor Bichen Wu, who has been a mentor to me and guided me through several research projects. Finally, I would like to thank my family for their unwavering support and encouragement.

Contrastive Learning for Context-Based Off-Policy Actor-Critic Reinforcement Learning

by Bernie Wang

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

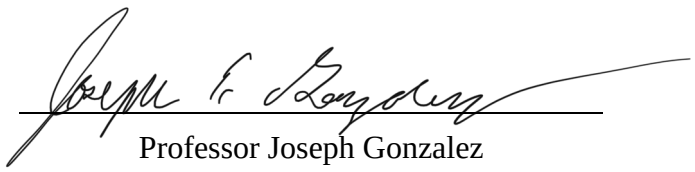
Committee:

Professor Kurt Keutzer
Research Advisor

December 17, 2020

(Date)

* * * * * * *



Professor Joseph Gonzalez
Second Reader

12/17/2020

(Date)

Abstract

Contrastive Learning for Context-Based Off-Policy Actor-Critic Reinforcement Learning

by

Bernie Wang

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Kurt Keutzer, Chair

Conventional reinforcement learning focuses on learning an optimal policy for a single task. Learning a large repertoire of tasks with this paradigm is inefficient in terms of the number of environment interactions. Meta-reinforcement learning aims to solve this problem by learning the underlying structure shared within a family of tasks. Recent work reformulates meta-reinforcement learning as contextual reinforcement learning where the contextual latent space is learned end-to-end. Representation of context is important and determines the ability of a policy to generalize. End-to-end reinforcement learning often struggle to generalize to unseen tasks because of the challenge of end-to-end policy optimization and representation learning. We introduce CoCOA: contrastive learning for context-based off-policy actor critic, which builds a contrastive learning framework on top of existing off-policy meta-RL. We evaluate CoCOA on a variety of continuous control and robotic manipulation tasks and show that adding a contrastive auxiliary task improves upon the policy returns and sample efficiency of end-to-end reinforcement learning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Reinforcement learning (RL) algorithms have shown great advances in single-task performance on simulated environments. In practice however, policies learned by single-task RL methods are in general not transferable to agents in the real world for the following reasons: (1) the dynamics of the real-world environment and agent may change over time or be different from their simulated counterparts, and (2) agents are often times required to perform multiple tasks.

Conventional RL approaches are limited to one policy per task, often requiring millions of interactions in the environment to learn an optimal policy. Learning a large repertoire of tasks by learning a policy per task quickly becomes unfeasible. Furthermore, single-task policies are sensitive to changes in the environment or within the agent, so a bag-of-policies approach cannot solve this problem.

Fortunately, tasks share common underlying structure that can be exploited. For example, placing a ball on a shelf and opening a door both involve grasping an object. A conventional RL algorithm would most likely fail to learn both tasks because the policy cannot differentiate these tasks. From an RL perspective, these two tasks differ solely in reward functions. If the agent can discern this difference, is it possible to learn a single policy for both tasks?

These shared structures among tasks have motivated research in meta-reinforcement learning (meta-RL), which focuses on learning to generalize to new tasks after interacting with the environment on a set of example tasks. Current meta-RL methods rely on encoding past experience into latent task representations with which they use to adapt their policies [6, 9, 24]. The standard approach to context-based meta-RL, shown in Figure 1.1, is to jointly train an encoder, which maps a history of state observations to a task embedding, and a policy, which maps observations to actions.

Although these methods have shown incredible improvements in sample efficiency compared to gradient-based meta-RL [10, 26], they struggle to generalize in environments with complex reward functions. Generalization is a challenge for end-to-end RL which attempts to learn an optimal encoder and an optimal policy at the same time [23]. As RL training procedures become increasingly more complex, policies that are trained end-to-end tend to

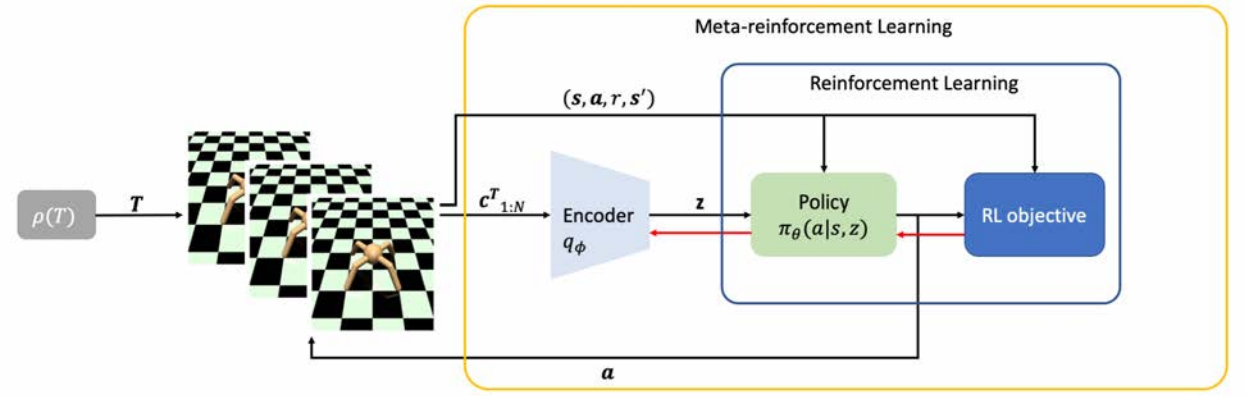


Figure 1.1: **Standard Meta-RL Approach:** In the standard meta-RL formulation, the agent interacts with an environment whose reward function or dynamics change based on the task $\mathcal{T}$ at hand. The encoder q_ψ maps context $\mathbf{c}^{\mathcal{T}}_{1:N}$ to task embedding $\mathbf{z}$, on which the policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ is conditioned. The arrows represent end-to-end gradients. In meta-RL, and encoder and policy are trained end-to-end.

generalize poorly to new unseen tasks.

In this work, we show that meta-training sample efficiency and meta-testing policy returns can be improved by adding an auxiliary contrastive prediction task to the meta-training procedure. Contrastive learning has been successful in learning differences and similarities among visual representations, and we argue that contrastive learning fits in well with the contextual meta-RL framework which aims to learn differences between past experience.

Our main contributions is CoCOA, contrastive learning for context-based actor-critic RL. Specifically, we define a contrastive framework by a discriminative objective, data augmentation for generating similar and dissimilar pairs, and similarity metric. Our method minimally modifies an existing context-based RL algorithm PEARL by adding an auxiliary loss during the batch update. We extensively evaluation our method against current meta-RL methods on a wide variety of continuous controls and robotic object manipulation tasks. We find that joint meta-learning and contrastive optimization improves upon end-to-end RL and outperforms existing methods.

Chapter 2

Related Works

Meta-learning was originally based on the idea of learning a parameter initialization that can quickly be optimized for a particular task. Early work proposed deploying base learners to fit to a particular task and having a meta-learner learn from the base learners to produce a better base learner [3, 28, 34]. [2], [14], and [17] applied this idea to learning to optimize deep neural networks. Meta-learning includes few-shot learning. Few-shot learning methods have been used for tasks like image classification [10, 33, 36] and generative modeling [8, 25]. Meta-learning in the context of reinforcement learning aims to learn policies [6, 10, 19] and dynamics models [20, 27] that can quickly adapt to unseen tasks.

Meta-RL algorithms that are based on the idea of learning an optimal parameter initialization [10, 26] learn a policy initialization that can attain single-task-level performance on a new task after one or few policy gradient steps. However, they suffer from poor meta-training and meta-testing sample efficiency because they are on-policy and leveraged gradient-based adaptations. Studies have shown that long-range temporal dependencies make learning a base learner difficult [21] while hyper-parameter sensitivity inhibits meta-learning from base learners [13].

Recent meta-RL methods learn from past experience by leveraging context, which is formulated as a collection of past agent-environment interactions. Recurrent and recursive meta-RL methods [6, 37] first demonstrated context-based meta-RL by aggregating experience into a latent representation on which their policy is conditioned. A policy neural network is trained to take experience in the form of context as an input by either augmenting the observation space with context or embedding context as the hidden state. Non-recurrent methods like [24] learn separate encoder networks that produce latent context variables. Adaptation for context-based meta-RL involves a single forward pass which is significantly faster than gradient-based optimization. More recent context-based meta-RL methods like [9] and [24] significantly improve sample efficiency by building context on top of off-policy RL algorithms.

Context has shown to be a key component to fast and sample-efficient meta-RL. In a study of context-based meta-RL, [9] found that a conventional RL algorithm trained with context to optimize the multi-task objective can rival state-of-the-art meta-RL algorithms. Although

context is powerful, it is only a meta-training technique. Context does not involve updating the policy but instead only augments its input space. The trade-off for fast adaptation via context inference is the lack of policy optimization on new tasks. Context-based meta-RL methods generally struggle to extrapolate to out-of-distribution tasks at meta-test time [18]. Policies conditioned on context suffer from generalization because they heavily rely on predicting the right task representation at test-time to adapt to new tasks.

We believe that learning a richer task representation by jointly optimizing an unsupervised objective with the meta-learning objective can improve the performance of context-based meta-RL. Prior work that used auxiliary self-supervised tasks include predicting the future given past experience [15, 22] or reconstruction tasks [32] to improve sample efficiency and performance of end-to-end RL. Contrastive learning has been used in RL to extract reward functions [31] and learn representations of visual inputs [7, 16]. In our work, we use contrastive learning to learn representations of context.

The importance of learning low-dimensional task embeddings from a sequence of past experience in meta-RL makes it well-formed for contrastive learning. Contrastive learning can be interpreted as a dictionary lookup task: given a query, find a positive key from a set of negatives [12]. The form of contrastive learning that our method uses is instance discrimination [4, 5, 38], where a set of queries and keys are generated from instances. A query and key are a positive pair if they are data augmentations of the same instance and negative otherwise. Our method applies contrastive learning on meta-RL by learning task representations that obey similarity constraints.

Chapter 3

Preliminaries

3.1 Meta-RL Problem Statement

We assume a distribution of tasks $\rho(\mathcal{T})$, where each task is a partially-observable Markov decision process (MDP), defined by the tuple

$$\mathcal{T} = (p(\mathbf{s}_0), p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t), r(\mathbf{s}_t, \mathbf{a}_t), \gamma, T), \tag{3.1}$$

where $\gamma \in [0, 1]$, $\mathbf{a}_t \in A \subset \mathbb{R}^d$, $\mathbf{s}_t \in S \subset \mathbb{R}^p$. Here, A is the action space, S is the state space, $p(\mathbf{s}_0)$ is the initial state distribution, $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ is the transition distribution, $r(\mathbf{s}_t, \mathbf{a}_t)$ is the reward function, γ is the discount factor, and T is the time horizon. The transition and reward functions are unknown, but can be sampled by taking actions in the environment. At timestep t, the agent takes an action $\mathbf{a}_t$ sampled from policy distribution $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$, and transitions to the next state $\mathbf{s}_{t+1}$ sampled from $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, collecting scalar reward $r(\mathbf{s}_t, \mathbf{a}_t)$. We define a trajectory as a sequence of transition tuples

$$\tau = \{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}'_t)\}_{t\in[0,T-1]}, \tag{3.2}$$

where $\mathbf{s}'_t = \mathbf{s}_{t+1}$.

We assume that all tasks in $\rho(\mathcal{T})$ share the same S, A, γ, T. Tasks within a distribution can vary in reward functions (e.g., moving objects to different target locations) and in transition functions (e.g., robots with different dynamics). The objective of an agent is to maximize the sum of discounted rewards for all tasks,

$$\mathbb{E}_{\mathcal{T}\in\rho(\mathcal{T})}[\mathbb{E}_{\tau\sim\pi_{\theta(\mathcal{T})}}[\textstyle\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)]], \tag{3.3}$$

where the inner expectation is taken on the trajectories sampled with a policy $\pi_{\theta(\mathcal{T})}$ adapted to task $\mathcal{T}$. The meta-training procedure optimizes the policy on a set of training tasks $\mathbf{D}_{train}$ such that

$$\theta_{meta} = \arg\max_\theta \mathbb{E}_{\mathcal{T}\in D_{train}}[l^{\mathcal{T}}_{meta}(\theta)], \tag{3.4}$$

where $l^{\mathcal{T}}_{meta}(\theta)$ is the algorithm-dependent meta-training loss for task $\mathcal{T}$.

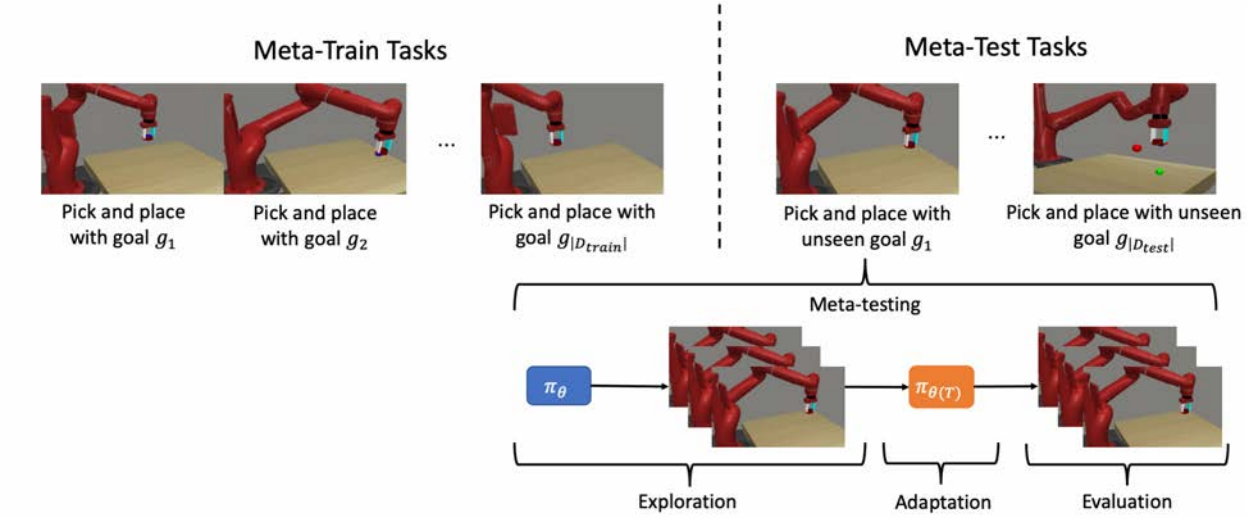


Figure 3.1: **Meta-RL evaluation protocol:** Meta-RL algorithms are train on a set of training tasks $\mathbf{D}_{train}$ and evaluated on a held out set of test tasks $\mathbf{D}_{test}$. Each task is sampled from $\rho(\mathcal{T})$ and uniquely corresponds to a specific goal state or system dynamics. The tasks are initialized before meta-training and remain constant throughout the meta-training and testing procedure. During meta-testing, the agent can explore the environment before adapting. The adapted policy is then evaluated on the same task.

The meta-testing performance is evaluated on a set of held out test tasks $\mathbf{D}_{test}$. Figure 3.1 shows how task sets are sampled for meta-RL. The agent can explore the environment for each task before adapting its policy to the task at hand. Typically, the sum of rewards averaged over all tasks evaluated on the adapted policy is reported as the meta-testing performance. Sample efficiency is essential in meta-RL, both in terms of the number of environment interactions during meta-training, and in the amount of experience required to adapt to a new task.

3.2 Context-based Meta-RL

The meta-training process learns a policy that adapts to the task at hand by conditioning on the history of past transitions, which we refer to as context $\mathbf{c}$. We define context as a sequence of transition tuples sampled from task $\mathcal{T}$:

$$c^{\mathcal{T}}_{1:N} = \{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')_n\}_{n\in[1,N]} \tag{3.5}$$

We denote the transition tuples by the subscript n because they are not necessarily ordered by time-step.

An encoder $f_\phi(\mathbf{c})$ maps context $\mathbf{c}_{1:N}^{\mathcal{T}}$ to context embedding (or task embedding) $\mathbf{z}$, on which the policy is conditioned as $\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z})$. Meta-training procedure jointly trains the policy and encoder on a variety of training tasks to infer the value of $\mathbf{z}$ from a recent history of experience in the new task.

3.3 Probabilistic Embeddings for Actor-Critic RL (PEARL)

PEARL is a context-based meta-RL algorithm built on top of the state-of-the-art off-policy Soft Actor-Critic [11] and jointly trains a probabilistic context encoder. The context embeddings are represented as normal distributions $\mathcal{N}(f_\phi^\mu(\mathbf{c}_{1:N}), f_\phi^\sigma(\mathbf{c}_{1:N}))$ and captures uncertainty about a task. The encoder is trained by back-propagating the gradient from the critic loss:

$$L_{\text{critic}} = \mathbb{E}_{\substack{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{B} \\ \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c})}}[Q_\theta(\mathbf{s},\mathbf{a},\mathbf{z}) - (r + \overline{V}(\mathbf{s}',\overline{\mathbf{z}}))]^2 \tag{3.6}$$

where $\mathcal{B}$ is the replay buffer, Q_θ is the critic Q network, $\overline{V}$ is the target value network, q_ϕ is the encoder network, and $\overline{\mathbf{z}}$ indicates no gradient is computed through it.

Modeling the context as probabilistic enables posterior sampling to explore the embedding space for less certain tasks. A key component for generalization in meta-RL methods is that meta-training inputs should match meta-testing inputs. That is, the embedding space on which posterior sampling is operated should match between meta-training and meta-testing. To accomplish this, in addition to the critic loss, the encoder also receives gradient from an information bottleneck:

$$\mathbb{E}_{\mathcal{T}}[\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c}^{\mathcal{T}})}[\beta D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{c}^{\mathcal{T}}) \parallel p(\mathbf{z}))]] \tag{3.7}$$

where $p(\mathbf{z})$ is a unit Gaussian prior over $\mathbf{z}$. The KL divergence term can be understood as an approximate bottleneck [1] that reduces overfitting to training tasks by constraining $\mathbf{z}$ to contain only the most essential information from the context to adapt to the task at hand.

The policy is trained to optimize the actor loss, which minimally modifies SAC's actor loss by conditioning the policy on $\mathbf{z}$:

$$L_{\text{actor}} = \mathbb{E}_{\substack{\mathbf{s}\sim\mathcal{B},\mathbf{a}\sim\pi_\theta \\ \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c})}}[\alpha \log \pi_\theta(\mathbf{a}|\mathbf{s}) - Q_\theta(\mathbf{s},\mathbf{a},\overline{\mathbf{z}})] \tag{3.8}$$

where α is the temperature.

3.4 Contrastive Unsupervised Representation Learning

Given a query q and set of keys $K = \{k_0, k_1, ...\}$ where every q matches with a positive key k_+, the goal of contrastive learning is to maximize the similarity between q and k_+ relative

to any other pairing between q and $k \in K \setminus \{k_+\}$. Intuitively, representations are learned by optimizing an objective that pulls similar pairs together while pushing dissimilar pairs apart in a latent space. The query-key pairs can be understood as different views of the same instance.

Chapter 4

Methodology

CoCOA minimally modifies an existing context-based actor-critic algorithm by training the contrastive objective as an auxiliary loss during meta-training. To specify a contrastive learning objective, we need to define (1) discrimination object, (2) the data augmentation for generating query-key pairs, and (3) the similarity measure used between the query-key embeddings in the contrastive loss.

In our experiments, we build our framework on top of PEARL, which leverages probabilistic task embeddings for exploration via posterior sampling. The similarity measure between query-key embeddings is tailored to probabilistic embeddings. Although we demonstrate our framework on PEARL, it can be adapted to any other context-based meta-RL algorithm by modifying the similarity measure.

4.1 Architectural Overview

CoCOA uses instance discrimination as the unsupervised auxiliary task. Instance discrimination is performed on transformed trajectories, where positive query-key context pairs are augmentations of the same trajectory. We use the negative Bhattacharyya distance to measure the similarity between two normal distributions. Lastly, we use a momentum encoding procedure proposed in MoCo [12], which [16] found to be better for RL. An overview of the CoCOA architecture is shown in Figure 4.1.

4.2 Discrimination Objective

Designing an appropriate prediction task is essential for contrastive learning. Instance discrimination on trajectories is the simplest approach because a trajectory is the largest unit of transition tuples known to belong to a task. Using more than one trajectory as an instance for our discrimination objective is not always feasible in off-policy RL because task identity is often times unknown. Our discrimination objective can be understood as a tra-

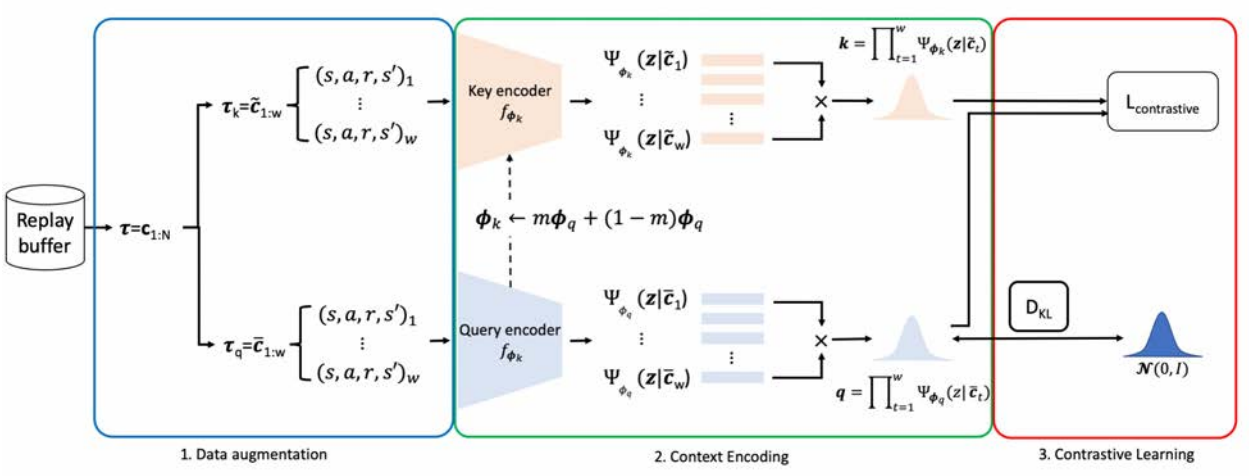


Figure 4.1: **Method Architecture**: (1) We generate a key τ_k and query τ_q pair by sampling a random temporal crop of size w from a trajectory τ. (2) The individual transition tuples in the key and query contexts are encoded by the key and query encoders f_{ϕ_k} and f_{ϕ_q} as Gaussian factors. The Gaussian factors are pooled (product) to obtain the key and query task embeddings $\mathbf{k}$ and $\mathbf{q}$. During the gradient update step, only the query encoder is updated. The key encoder weights are the moving average of the query weights. (3) The query encoder is optimized from the contrastive loss as well as an information bottleneck on the queries.

jectory identification task, where we maximize the agreement between two augmentations of a trajectory.

4.3 Query-Key Pair Generation

A significant difference between context-based meta-RL and conventional RL is that meta-RL operates on a trajectory of data instead of a single transition. A key factor of successful generalization is learning a rich task representation from a limited amount of experience.

We apply random temporal cropping on trajectories. That is, we sample a window of transition tuples from a trajectory. A positive query-key pair consists of two windows of the same trajectory while negative key-query pairs are windows of different trajectories. The size of the window must be sufficiently large in order for the keys and query to contain enough information to describe a task.

Since CoCOA is build on top of PEARL, our method encodes individual transition tuples as Gaussian factors. Subsequently, an encoded windowed trajectory is the product of

independent factors

$$f_\phi(\mathbf{z}|\mathbf{c}_{1:N}) \propto \prod_{t=1}^{N} \Psi_\phi(\mathbf{z}|\mathbf{c}_t), \tag{4.1}$$

where $\Psi_\phi(\mathbf{z}|\mathbf{c}_t) = N(\mu_\phi(\mathbf{c}_t), \sigma_\phi(\mathbf{c}_t))$ is a Gaussian factor.

For a batch of N trajectories, we generate N keys and N queries. We do not sample negative examples explicitly. Instead, given a positive key k_+ for query q, we treat the other $N-1$ keys within a batch as negative examples.

4.4 Similarity Measure

Another determining factor in the discrimination objective is the similarity function used to measure agreement between query-key pairs. Since task representations are modelled as probabilistic embeddings, we consider metrics that express similarities between normal distributions. Consider a query-key pair represented as normal distributions $k = \mathcal{N}(\mu_1, \Sigma_1)$ and $q = \mathcal{N}(\mu_2, \Sigma_2)$. We employ the similarity function:

$$\text{sim}(q, k) = -(\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)^T, \tag{4.2}$$

where $\Sigma = \frac{\Sigma_1 + \Sigma_2}{2}$. This function is based on the negative Bhattacharyya distance which measures the similarity between two normal distributions. Intuitively, the similarity between two normal distributions is high when their means are close together or when they have a large overlap, as shown by the inverse relationship between similarity and Σ, the average covariance matrix.

4.5 Contrastive Loss

We use the InfoNCE score function introduced by [22] coupled with our similarity measure to compute the contrastive loss:

$$L_{contrastive} = -\frac{1}{N} \sum_{i=0}^{N-1} \log \frac{\exp(\text{sim}(q_i, k_+)}{\sum_{j=0}^{N-1} \exp(\text{sim}(q_i, k_j))} \tag{4.3}$$

where k_+ is the positive key for a given query q. Given a query and batch of N keys, we treat the InfoNCE loss as an N-way cross-entropy loss where k_+ the label.

We found empirically that the contrastive loss pushes task embeddings further apart without bounds, causing the mean and variance of the embeddings to explode. This causes poor performance because of the distribution mismatch between the learned task embeddings and exploration embedding space. Recall that exploration is modelled by posterior sampling. It is crucial that our learned embeddings matches the latent exploration space, so we add a KL term $D_{\text{KL}}(q \parallel \mathcal{N}(0, I))$ to regularize the embeddings.

4.6 Implementation

CoCOA is built on top of PEARL, which trains encoder f_ϕ and policy π_θ end-to-end. To adapt the PEARL architecture to CoCOA, we jointly optimize the critic loss (Equation 3.6) and information bottleneck (Equation 3.7) with our contrastive loss (Equation 4.3) during the batch update.

We use a query encoder f_q for encoding queries and a key encoder f_k for encoding keys. The query encoder is the same encoder as f_ϕ, which is updated by back-propagating the joint critic-contrastive loss. The key encoder is an exponentially moving average version of the query encoder, a method proposed by [12]. Given the query encoder parameterized by θ_q, key encoder parameterized by θ_k, and momentum m, the update equation for the key encoder is

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q \tag{4.4}$$

Chapter 5

Experiments

In our experiments, we evaluate how CoCOA compares to current meta-RL methods, in terms of the test-time average return, on two meta-RL benchmarks that are both simulated via the MuJoCo simulator [35]: (1) MuJoCo Controls benchmark: a set of six continuous control environments focused around robotic locomotion, and (2) Metaworld [39]: a family of robotic manipulation tasks with everyday objects that all share the same table-top environment with a simulated Sawyer arm.

5.1 Baselines

For both benchmarks, we evaluate the performance and sample efficiency of CoCOA to PEARL as well as existing policy gradient meta-RL methods MAML with TRPO [10, 30] and RL2 with PPO [6, 29] using publicly available code. RL2 is an recurrence-based policy gradient method that implements its task embedding as the hidden state of its recurrent policy network.

To demonstrate the difficulty of the tasks, we also implement a goal-conditioned SAC trained on an oracle task encoder. Instead of learning task embeddings, the policy is con-

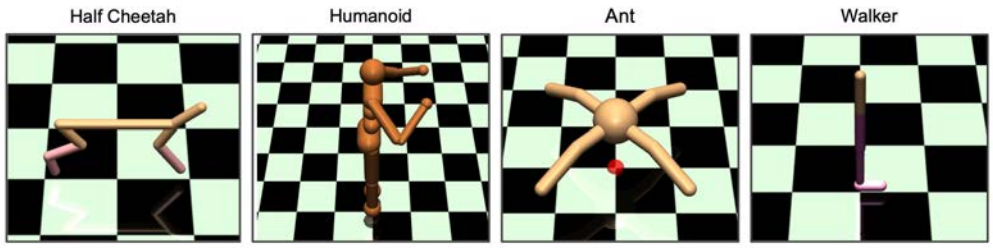


Figure 5.1: Simulation of MuJoCo Controls agents

ditioned on hand-designed features pertaining to the task at hand. For example, the task embedding for a 2D navigation task would be the Cartesian coordinates of the target location. This oracle-SAC serves as an empirical upper bound for context-based actor-critic RL. The results of each algorithm are averaged across three random seeds.

5.2 Setup and Results - MuJoCo Controls

Environment Setup

The MuJoCo Controls benchmark consists of six continuous control environments focused on robotic locomotion. It was first introduced by [10] and [26] and has since been the de facto standard in meta-RL benchmarks [9, 24]. Each environment is characterized by a robotic agent and a family of locomotion tasks. Tasks within a family may differ in reward functions (walking direction for Half-Cheetah-Forward-Backward, Ant-Forward-Backward, Humanoid-Dir-2D; target velocity for Half-Cheetah-Vel; goal location for Ant-Goal-2D) or agent dynamics (random agent system parameters for Walker-2D-Rand-Params). Figure 5.1 depicts the agents involved in the MuJoCo Controls benchmark. All tasks have horizon length 200.

Evaluation

Meta-RL algorithms are evaluated on how well the agent can distinguish and adapt to unseen test tasks. To evaluate on meta-testing tasks, we allow all methods two exploration trajectories to adapt their policy to the task at hand. The meta-testing performance is the average returns of trajectories collected after each method has adapted. In order to evaluate both meta-testing performance and sample efficiency, we report meta-testing average returns as a function of the number of environment samples collected during meta-training.

Results

The meta-testing results for MuJoCo Controls are shown in Figure 5.2. We note that the Ant-Goal performance for PEARL is referenced from [24], as we were not able to reproduce their results with their implementation. We suspect that is the reason why CoCOA, which is build on top of PEARL, would also under-perform on Ant-Goal. For all environments except Ant-Goal, our method achieves similar or better meta-testing performance than the baselines. The significant performance gap between the SAC-based methods and the on-policy MAML and RL2 highlights the importance of off-policy RL for sample efficiency.

We are surprised by the performance of our oracle-SAC baseline. In 4 out of 6 environments, oracle-SAC performs similarly or worse than PEARL and CoCOA. We hypothesize that the poor Walker-2D-Rand-Params performance of oracle-SAC relative to PEARL and CoCOA is attributed to the high dimensionality of the hand-crafted task embedding (67 di-

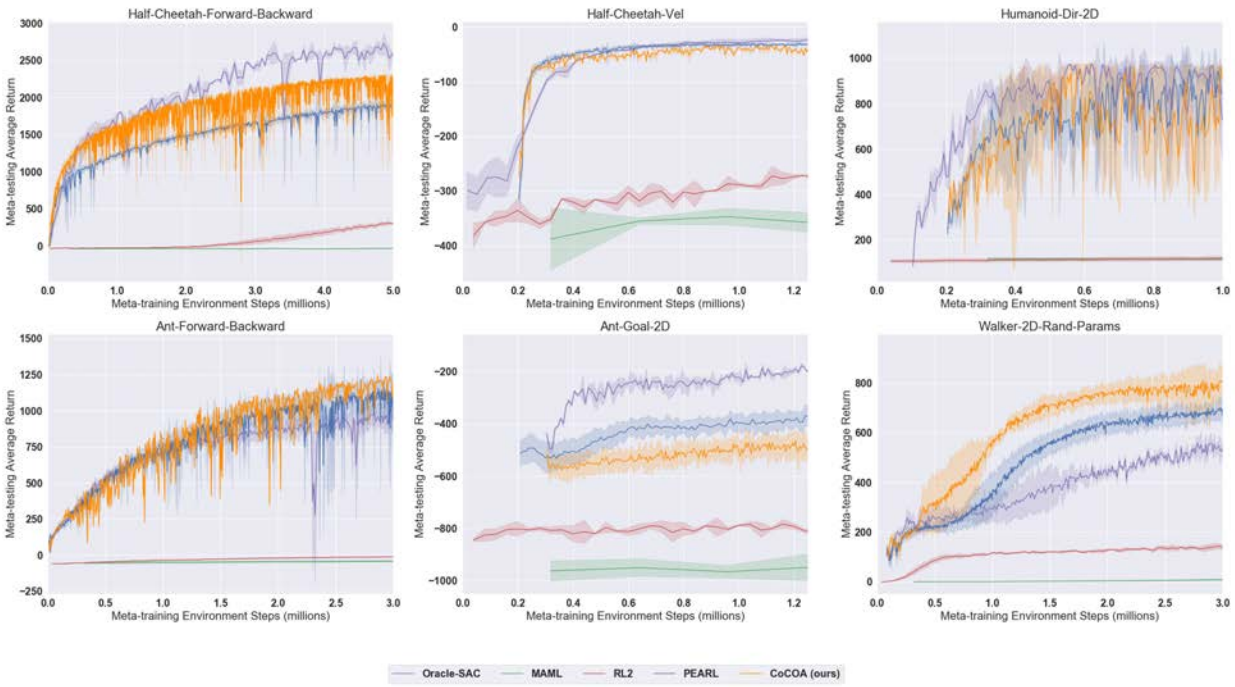


Figure 5.2: **MuJoCo Controls benchmark:** Meta-test average returns vs. environment samples collected during meta-training. Our method performs comparably or better than previous meta-RL methods on 5 of the 6 environments. The performance of PEARL on Ant-Goal was reported by [24], but we are not able to reproduce the results.

mensions vs. 5 dimensions for a learned embedding), where each dimension corresponds to a random system parameter. This highlights the importance of constraining task embeddings with an information bottleneck such that the policy receives only the most important task information.

The underwhelming performance of oracle-SAC suggests that there is not much room for improvement in representation learning in meta-RL. Two out of the six environments are binary walking tasks which are relatively trivial. Walker and humanoid are particular difficult agents to control, even for single-task benchmarks. We hypothesize that policy optimization is the performance bottleneck for these environments,

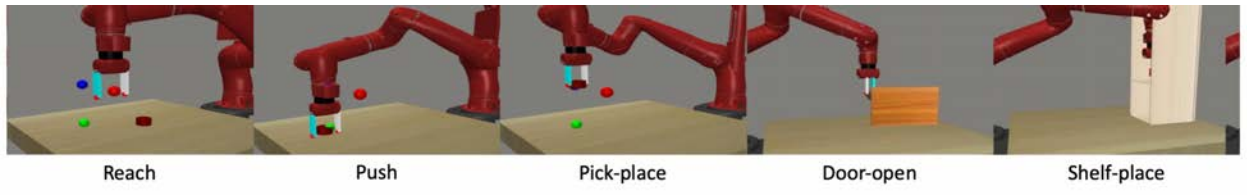


Figure 5.3: Examples of Metaworld environments simulated in MuJoCo. All environments share the same table-top setting and Sawyer robotic arm.

5.3 Setup and Results - Metaworld

Environment Setup

Metaworld consists of a family of object manipulation tasks that all share the same robot arm on a table-top setup, shown in Figure 5.3. The robot arm is a 7-DOF Sawyer robot that is simulated via MuJoCo, with the action space corresponding to the velocity of the 3D end-effector and the control of the gripper.

We report results on 30 environments from this benchmark, ranging from opening a door at random positions to placing a puck onto randomly located shelves. A task for a particular environment corresponds to a random initial object and goal positions.

Evaluation

We follow the same evaluation procedures used for MuJoCo Control. Methods are allowed 10 exploration trajectories per task for adaptation. All of the tasks have horizon 200. Each environment has 50 meta-training tasks and 10 held out meta-testing tasks. We limit the number of meta-training environment samples to 2.5 million timesteps, as we saw marginal performance gains after this threshold.

Results

We evaluate our method against the baselines on 30 tasks and report it in Table 5.1. CoCOA achieves the highest average test-time return in 15 out of 30 environments. When comparing against PEARL, the base algorithm, our method outperforms it in 20 out of 30. We found RL2's performance to be surprising in that unlike in MuJoCo Controls, RL2 outperforms the other methods on a substantial amount of environments. On-policy methods typically suffer from sample efficiency but benefit from stability.

Training curves for Reach, Push, and Pick-Place environments are shown in Figure 5.4. We also include the learning curves for the oracle baseline. We found CoCOA to be somewhere between PEARL and oracle-SAC and note that the gap between oracle-SAC and

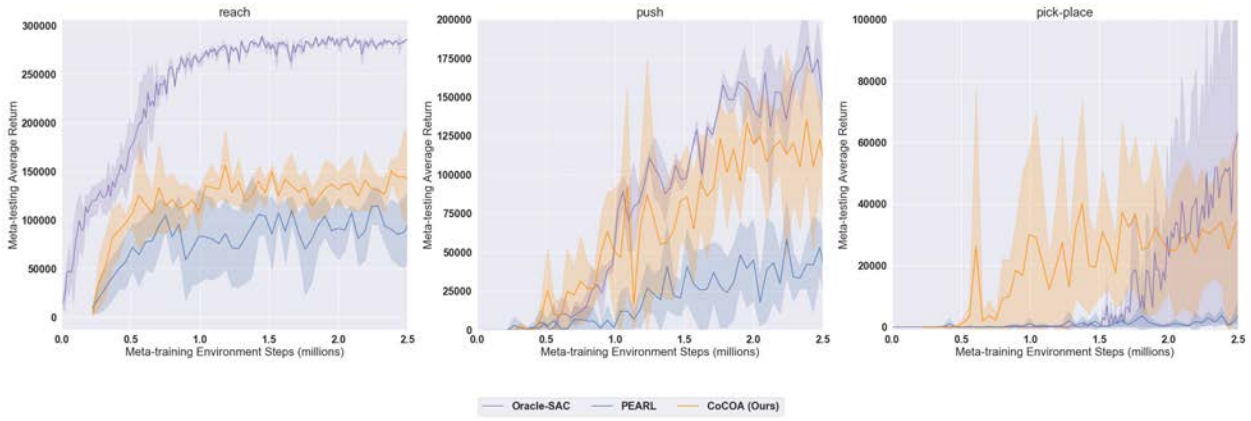


Figure 5.4: **Metaworld Reach, Push, Pick-place environments:** These environments are the standard robot manipulation tasks that have been modified for the Metaworld benchmark. Plotted are meta-test average returns vs. environment samples collected during meta-training. CoCOA performs better than previous meta-RL methods on all three of the environments.

PEARL is significantly larger in Metaworld than in MuJoCo Controls. We argue that the joint controls in MuJoCo Controls are harder to learn than the end effector controls in Metaworld benchmark. On the other hand, the reward functions in MetaWorld are more complex than those in MuJoCo Controls. Since our method focuses on learning task representations, we see more improvements on Metaworld than on MuJoCo Control.

These findings seem to support the hypothesis that meta-learning task can be attributed to two challenges: (1) the actual dynamics of task at hand (policy optimization) and (2) learning to distinguish tasks (representation learning).

Ablations

We include an ablation on the importance of similarity measure for probabilistic task embeddings. Table 5.2 compares the meta-test time performance betweeen versions of CoCOA trained with the Bhattacharyya-based similarity measure against bilinear inner product $\text{sim}(q, k) = q^T W k$, which [12, 16, 22] found to outperform normalized cosine similarity. The projection matrix W is a learned parameter.

Environment	MAML	RL2	PEARL	CoCOA (ours)
Sweep	2391.0	**19596.5**	3948.6	7896.7
Sweep-into	8564.3	**33741.4**	26129.5	5025.0
Push-back	-30.8	47883.2	69111.3	**73401.0**
Dial-turn	12408.9	27538.7	4661.1	**53383.6**
Coffee-button	11096.7	**122752.1**	2504.1	-87.1
Assembly	**3238.0**	-13.8	-40.8	122.4
Pick-out-of-hole	-40.2	-31.0	-18.3	**-17.8**
Shelf-place	-31.9	-20.4	**14010.4**	456.6
Handle-press-side	6870.8	**71913.6**	11824.3	-83.0
Plate-slide	5007.6	37721.4	38278.6	**71191.4**
Button-press-wall	22614.0	**181956.3**	-83.3	235.8
Handle-pull	6109.3	**169778.4**	2496.2	-76.0
Plate-slide-back-side	573.9	**30637.0**	13665.2	3019.6
Drawer-close	680.1	23054.6	48095.5	**66591.6**
Reach	43680.4	84190.4	90500.4	**142177.1**
Reach-wall	45484.4	108727.8	164820.7	**196282.0**
Push	6810.2	21780.1	45466.1	**118123.4**
Pick-place-wall	-34.7	699.4	11483.9	**54632.4**
Pick-place	2418.4	-23.1	1881.9	**30804.1**
Peg-unplug-side	10368.6	111051.1	**230571.66**	202572.6
Window-open	5464.3	**15353.5**	-98.6	-11499.7
Door-close	962.3	**46638.2**	6643.0	1824.5
Hand-insert	64530.1	208022.1	325252.1	**333845.5**
Door-lock	688.2	58862.6	152948.8	**217496.0**
Bin-picking	-47.4	-46.2	44.9	**248809.9**
Soccer	5636.3	31799.6	38138.9	**153006.8**
Push-wall	3262.0	7537.0	66786.9	**112551.1**
Coffee-pull	-117.0	5389.7	**64322.8**	58043.2
Button-press-topdown-wall	-74.6	**280400.5**	-92.3	-75.4
Faucet-close	-124.4	**36485.7**	4180.0	8600.5

Table 5.1: **Metaworld Benchmark:** Meta-test average returns achieved at 2.5 million time-steps. CoCOA performs better than previous meta-RL methods on all **15** out of **30** environments. Compared to its base algorithm PEARL, our method achieves higher meta-test returns on **20** out of **30** environments.

Environment	Bhattacharyya	Bilinear
Button-press-topdown-wall	-75.4	**2149.5**
Coffee-pull	**58043.2**	56147.6
Dial-turn	**38541.5**	23214.1
Door-close	**6059.8**	5266.7
Faucet-close	**8600.5**	7183.0
Pick-out-of-hole	**-18.1**	-18.6
Plate-slide-back-side	**16557.1**	10898.8
Push-wall	**112551.1**	64173.2
Soccer	**153006.8**	108311.7
Window-open	1434.7	**4996.7**

Table 5.2: **Similarity measure ablation experiment.** Meta-test average returns achieved at 2.5 million time-steps on 10 Metaworld environments. Our method with Bhattacharyya distance outperforms another version with bilinear product on **8** out of **10** environments.

Chapter 6

Conclusion

In this work, we presented CoCOA, contrastive learning for context-based off-policy actor critic. Our method involves building a framework for contrastive learning of task representations in meta-RL, which when built on top of an existing context-based method can significantly improve meta-testing performance and sample efficiency. We formulate the contrastive auxiliary task as a trajectory identification task, where positive query-key pairs are data augmentations of the same trajectory. The similarity between keys and queries are based on the overlap between the normal distributions that they represent. Although we demonstrate through our experiments that CoCOA can outperform existing methods on a diverse set of benchmark environments, methods like RL2 show better generalization on some environments due to the relative stability of on-policy algorithms. A promising direction for future work is to adapt the contrastive learning framework to on-policy methods.

Bibliography

[1] Alexander A. Alemi et al. *Deep Variational Information Bottleneck*. 2019. arXiv: 1612.00410 [cs.LG].

[2] Marcin Andrychowicz et al. *Learning to learn by gradient descent by gradient descent*. 2016. arXiv: 1606.04474 [cs.NE].

[3] Jonathan Baxter. "Learning internal representations". In: *Proceedings of the eighth annual conference on Computational learning theory - COLT '95* (1995). DOI: 10.1145/225298.225336. URL: http://dx.doi.org/10.1145/225298.225336.

[4] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. arXiv: 2002.05709 [cs.LG].

[5] Alexey Dosovitskiy et al. *Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks*. 2015. arXiv: 1406.6909 [cs.LG].

[6] Yan Duan et al. *RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning*. 2016. arXiv: 1611.02779 [cs.AI].

[7] Debidatta Dwibedi et al. *Learning Actionable Representations from Visual Observations*. 2019. arXiv: 1808.00928 [cs.CV].

[8] Harrison Edwards and Amos Storkey. *Towards a Neural Statistician*. 2017. arXiv: 1606.02185 [stat.ML].

[9] Rasool Fakoor et al. *Meta-Q-Learning*. 2020. arXiv: 1910.00125 [cs.LG].

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. 2017. arXiv: 1703.03400 [cs.LG].

[11] Tuomas Haarnoja et al. *Soft Actor-Critic Algorithms and Applications*. 2019. arXiv: 1812.05905 [cs.LG].

[12] Kaiming He et al. *Momentum Contrast for Unsupervised Visual Representation Learning*. 2020. arXiv: 1911.05722 [cs.CV].

[13] Peter Henderson et al. *Deep Reinforcement Learning that Matters*. 2019. arXiv: 1709.06560 [cs.LG].

[14] Sepp Hochreiter, Arthur Younger, and Peter Conwell. "Learning To Learn Using Gradient Descent". In: Sept. 2001, pp. 87–94. ISBN: 978-3-540-42486-4. DOI: 10.1007/3-540-44668-0_13.

[15] Max Jaderberg et al. "Human-level performance in 3D multiplayer games with population-based reinforcement learning". In: *Science* 364.6443 (May 2019), pp. 859–865. ISSN: 1095-9203. DOI: 10.1126/science.aau6249. URL: http://dx.doi.org/10.1126/science.aau6249.

[16] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. "CURL: Contrastive Unsupervised Representations for Reinforcement Learning". In: *Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119* (2020). arXiv:2004.04136.

[17] Ke Li and Jitendra Malik. *Learning to Optimize*. 2016. arXiv: 1606.01885 [cs.LG].

[18] Russell Mendonca et al. *Meta-Reinforcement Learning Robust to Distributional Shift via Model Identification and Experience Relabeling*. 2020. arXiv: 2006.07178 [cs.LG].

[19] Nikhil Mishra et al. *A Simple Neural Attentive Meta-Learner*. 2017. arXiv: 1707.03141 [cs.AI].

[20] Anusha Nagabandi et al. *Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning*. 2019. arXiv: 1803.11347 [cs.LG].

[21] Alex Nichol, Joshua Achiam, and John Schulman. *On First-Order Meta-Learning Algorithms*. 2018. arXiv: 1803.02999 [cs.LG].

[22] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. *Representation Learning with Contrastive Predictive Coding*. 2019. arXiv: 1807.03748 [cs.LG].

[23] Charles Packer et al. *Assessing Generalization in Deep Reinforcement Learning*. 2019. arXiv: 1810.12282 [cs.LG].

[24] Kate Rakelly et al. *Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables*. 2019. arXiv: 1903.08254 [cs.LG].

[25] Danilo Jimenez Rezende et al. *One-Shot Generalization in Deep Generative Models*. 2016. arXiv: 1603.05106 [stat.ML].

[26] Jonas Rothfuss et al. *ProMP: Proximal Meta-Policy Search*. 2018. arXiv: 1810.06784 [cs.LG].

[27] Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. *Meta Reinforcement Learning with Latent Variable Gaussian Processes*. 2018. arXiv: 1803.07551 [stat.ML].

[28] Jürgen Schmidhuber. *Evolutionary Principles in Self-referential Learning: On Learning how to Learn: the Meta-meta-meta...-hook*. 1987.

[29] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].

[30] John Schulman et al. *Trust Region Policy Optimization*. 2017. arXiv: 1502.05477 [cs.LG].

[31] Pierre Sermanet et al. *Time-Contrastive Networks: Self-Supervised Learning from Video*. 2018. arXiv: 1704.06888 [cs.CV].

[32] Evan Shelhamer et al. *Loss is its own Reward: Self-Supervision for Reinforcement Learning*. 2017. arXiv: 1612.07307 [cs.LG].

[33] Jake Snell, Kevin Swersky, and Richard S. Zemel. *Prototypical Networks for Few-shot Learning*. 2017. arXiv: 1703.05175 [cs.LG].

[34] Sebastian Thrun and Lorien Pratt, eds. *Learning to Learn*. USA: Kluwer Academic Publishers, 1998. ISBN: 0792380479.

[35] E. Todorov, T. Erez, and Y. Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033.

[36] Oriol Vinyals et al. *Matching Networks for One Shot Learning*. 2017. arXiv: 1606.04080 [cs.LG].

[37] Jane X Wang et al. *Learning to reinforcement learn*. 2016. arXiv: 1611.05763 [cs.LG].

[38] Zhirong Wu et al. *Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination*. 2018. arXiv: 1805.01978 [cs.CV].

[39] Tianhe Yu et al. *Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning*. 2019. arXiv: 1910.10897 [cs.LG].