Learning Context-aware Task Reasoning for Efficient Meta-reinforcement Learning

Haozhe Wang
wanghzh@shanghaitech.edu.cn
ShanghaiTech University
Shanghai, china

Jiale Zhou
zhoujl@shanghaitech.edu.cn
ShanghaiTech University
Shanghai, china

Xuming He
hexm@shanghaitech.edu.cn
ShanghaiTech University
Shanghai, china

ABSTRACT

Despite recent success of deep network-based Reinforcement Learning (RL), it remains elusive to achieve human-level efficiency in learning novel tasks. While previous efforts attempt to address this challenge using meta-learning strategies, they typically suffer from sampling inefficiency with on-policy RL algorithms or meta-overfitting with off-policy learning. In this work, we propose a novel meta-RL strategy to address those limitations. In particular, we decompose the meta-RL problem into three sub-tasks, task-exploration, task-inference and task-fulfillment, instantiated with two deep network agents and a task encoder. During meta-training, our method learns a task-conditioned actor network for task-fulfillment, an explorer network with a self-supervised reward shaping that encourages task-informative experiences in task-exploration, and a context-aware graph-based task encoder for task inference. We validate our approach with extensive experiments on several public benchmarks and the results show that our algorithm effectively performs exploration for task inference, improves sample efficiency during both training and testing, and mitigates the meta-overfitting problem.

KEYWORDS

Multitask Learning; Deep Reinforcement Learning

1 INTRODUCTION

Modern reinforcement learning has achieved great successes in solving certain complex tasks by utilizing deep neural networks, which can even be trained from scratch [5, 28, 36]. Such successes, however, require a large amount of training experiences for new tasks. In contrast, human learners are able to exploit past experiences when facing a novel problem and quickly learn skills for a related task [17]. To achieve such fast adaptation is a critical step towards building a general AI agent capable of solving multiple tasks in real-world environments.

A principled way to tackling the problem of efficient adaptation is the meta learning framework [8], which aims to capture shared knowledge across tasks and hence enables an agent to learn a similar task using only a few experiences. In the reinforcement learning setting, however, as the learning agent also needs to explore in each novel task, it is particularly challenging to design an efficient meta-RL algorithm. A majority of prior works adopt on-policy RL algorithms [7, 8, 10, 26], which are data-inefficient during meta-training [25]. To remedy this, Rakelly et al. [25] propose an alternative strategy, PEARL, that relies on off-policy RL methods to achieve sample efficiency in meta-training. By introducing a latent variable to represent a task, their method decomposes the problem into online task inference and task-conditioned policy learning that uses experiences from a replay buffer (i.e., off-policy learning).

Nevertheless, such an off-policy strategy has several limitations during meta-test stage, particularly for the few-shot setting. First, it ignores the role of exploration in the task inference (cf. meta-episodes in [7]), which is critical in efficient adaptation as the exploration is responsible for collecting informative experiences within the few episodes to infer tasks. In addition, the agent has to explore in an online fashion for task inference during meta-test, which typically has a different sample distribution from the replay buffer that provides adaptation data at meta-train stage. PEARL partially alleviates this problem of train-test mismatch [38] by adopting a replay buffer of recently collected data. Such an in-between strategy [25], however, still leads to severe "meta-overfitting", particularly for online task inference (cf. Sec 4.1&5.3). Furthermore, the adaptation data acquired by exploration are simply aggregated by a weighted average pooling for task inference. As the experience samples are not iid, such aggregation fails to capture their dependency relations, which is informative for task inference.

In this work, we propose a context-aware task reasoning strategy for meta-reinforcement learning to address the aforementioned limitations. Adopting a latent representation for tasks, we formulate the meta-learning as a POMDP, which learns an approximate posterior distribution of the latent task variable jointly with a task-dependent action policy that maximizes the expected return for each task. Our main focus is to develop an adaptive task inference strategy that is able to effectively map from few-shot experiences of a task to its representation without suffering from the meta-overfitting.

To achieve this, we design a novel task inference network that consists of an exploration policy network and a structured task encoder shared by all tasks. Learning an exploration policy allows us to explore a task environment more efficiently and to introduce regularization to bridge the gap between meta-train and meta-test, leading to better generalization. The structured task encoder is built on a context-aware graph network, which is input permutation-invariant and size-agnostic in order to cope with variable number of exploration episodes. Our graph network encodes task-related

dependency in experience data samples, capable of capturing complex task statistics from exploration to achieve more data-efficient task inference.

To train our meta-learner, we develop a variational EM formulation that alternately optimizes the exploration policy network that collects experiences for a task, the context-aware graph network that performs task inference based on the collected task data, and an action policy network that aims to complete tasks towards maximum rewards given the inferred task information. Our meta-learning objective, formulated under the POMDP framework, is composed of two state-action value functions for the two respective policies. The state-action value for the exploration policy is guided by a reward shaping strategy that encourages the policy to collect task-informative experiences, and is learned with Soft Actor Critic (SAC) [11] for randomized behavior to narrow the gap between online rollouts and offline buffers. In essence, our method decomposes the meta-RL problem into three sub-tasks, task-exploration, task-inference and task-fulfillment, in which we learn two separate policies for different purposes and a task encoder for task inference.

We evaluate our meta-reinforcement learning framework on four benchmarks with a diverse task distributions, in which our approach outperforms prior works by improving training efficiency (up to 400x) and testing efficiency with better asymptotic performance (up to 300%) while effectively mitigating the meta-overfitting by a large margin (up to 75%). Our contributions can be summarized as follows:

- We propose a sample-efficient meta-RL algorithm that achieves the state-of-the-art performances on multiple benchmarks with diverse task distributions.
- We present a dual-agents design with a reward shaping strategy that explicitly optimizes for the ability for task exploration and mitigates meta-overfitting.
- We develop a context-aware graph network for task inference that models the dependency relations between experience data in order to efficiently infer task posterior.

2 RELATED WORK

2.1 Meta-reinforcement Learning

Prior meta reinforcement learning methods can be categorized into the following three lines of work.

The first line of work adopts a learning-to-learn strategy [8, 10, 26, 30, 41]. In particular, MAML [8] meta-learns a model initialization from which to adapt the parameters with policy gradient methods. To tackle issues in computing second-order derivatives for MAML, ProMP [26] further proposes a low-variance curvature estimator to perform proximal policy optimization that bounds the policy changeRecently, MAESN [10] improves MAML with more temporally coherent exploration by injecting structured noise via a latent variable conditioned policy, and enables fast learning of exploration strategies.

The second category of approaches uses recurrent or recursive models to directly meta-learn a function that takes experiences as input and generates a policy for an agent [7, 21, 39]. Among them, RL^2 [7] trains a recurrent agent with on-policy meta-episodes that comprise a sequence of exploration and task-fulfillment episodes, which aims to maximize the expected return of the task-fulfillment episode. Their method essentially learns to extract task information from the first few rounds of exploration, encoded in the latent states of its RNN, and complete the task in the final episode based on the known task information (latent states).

The first two groups of work often learn a single policy to perform exploration for policy adaptation and task-fulfillment, which overloads the agent with two distinct objectives. By contrast, we learn two separate policies, one focusing on exploration for policy adaptation and the other for task-fulfillment.

In the third line of work, Rakelly et al. [25] and Humplik et al. [14] propose to first infer task with task experiences and adapts the agent according to the task knowledge. Framing meta-RL as a POMDP problem with probabilistic inference, Humplik et al. [14] formulate task inference as belief state in POMDP, and learn the inference procedure in a supervised manner with privileged information as ground truth. PEARL [25] learns to infer tasks in an unsupervised manner by introducing an extra reconstruction term and incorporates off-policy learning to improve sample efficiency. However, both approaches ignore the role of exploration in task inference, and PEARL suffers from train-test mismatch in the data distribution for task inference.

By contrast, we propose to further disentangle meta-RL into task-exploration, task-inference and task-fulfillment, and introduce an exploration policy within a variational EM formulation. Our method enhances the inference procedure via active task exploration and effective task inference, achieves data-efficiency both in meta-train and meta-test, and mitigates meta-overfitting.

2.2 Relational Modeling on Sets

The problem of inference on a set of samples has been widely explored in literature [2, 6, 9, 12, 15, 19], and here we focus on those approaches that learn a permutation-invariant and input size-agnostic model.

DeepSets [42] proposes a general design principles for permutation invariant functions on sets. As an instantiation, Sets2Sets [37] encodes a set using an attention mechanism, which retrieves vectors in a manner immune to the shuffle of memory and accepts variable size of inputs. However, they typically do not consider relations among input elements, which is critical for modeling short experiences for task inference.

Self-attention module [1, 13, 27, 40] is commonly used to model pairwise relationships between entities of a set. Among them, Non-local Neural Networks [40] are capable of capturing global context using the scaled dot product attention [34]. Along this direction, graph networks [2, 9] provide a flexible framework that can model arbitrary relationships among entities and is invariant to graph isomorphism. Self-attention on sets, a special case of graphs, can naturally be assimilated into the graph network framework [2, 35, 43]. Recently, LatentGNN [43] develops a novel undirected graph model to encode non-local relationships through a shared latent space and admits efficient message passing due to its sparse connection. In this work, the design of our graph-based inference network is inspired by LatentGNN and DeepSets, aiming to efficiently model the dependency relationship in experience data.

3 VARIATIONAL META-REINFORCEMENT LEARNING

We aim to address the fast adaptation problem in reinforcement learning, which allows the learned agent to explore a new task (up to a budget) and adapts its policy to the new task. We adopt the meta-learning framework to enable fast learning, incorporating a prior of past experiences in structurally similar tasks.

Formally, we assume a task distribution $p(\mathcal{T})$ from which multiple tasks are i.i.d sampled for meta-training and testing. Any task $\mathcal{T} \sim p(\mathcal{T})$ can naturally be defined by its MDP $(S, A, \rho(s_0), P, \gamma, r)$ in RL, where S is the state space, A is the action space, $\rho(s_0)$ is the distribution of the initial states, $P(s'|s, a)$ is the transition distribution, γ is the discount in rewards which we will omit for clarity, and $r(s, a)$ is the reward function. Here the task distribution is typically defined by the distribution of the reward function and the transition function, which can vary due to some underlying parameters. We introduce a latent variable $z \sim p(z) = \mathcal{N}(0, 1)$ to represent the cause of task variations, and formulate the meta RL problem as a POMDP in which z serves as the unobserved part of the state space.

We first introduce two policies, including an action policy π that aims to maximize expected reward under a given task z and an exploration policy ξ for generating task-informative samples by interacting with the environment. We then adopt the off-policy Actor-Critic framework [29], and learn two Q-functions Q^π, Q^ξ for the action and exploration policy respectively. To achieve this, we define the following learning objective, which maximizes the expected log likelihood of returns under two policies (we omit the expectation over tasks for clarity):

$$\underset{\substack{\beta(s,a) \\ E(s',r|\cdot)}}{\mathbb{E}} \left[\log P(R = y^\pi | s, a) \right] + \underset{\substack{\rho(s,a) \\ E(s'|\cdot), r^\xi}}{\mathbb{E}} \left[\log P(R = y^\xi | s, a) \right] \quad (1)$$

where β, ρ denotes the (s, a) distribution visited by the two behavior policies for π, ξ, E denotes the environment, and r^ξ is the reshaped reward for ξ (Sec. 4.1). Typically, β, ρ is approximated by the experience replay buffers of π, ξ, denoted as $\mathcal{B}, \mathcal{X}$ respectively [11, 20, 22]. We will slightly abuse the notation β, ρ to denote the joint distribution of (s, a, s', r) for brievity.

Here $P(R|s, a)$ denotes the likelihood of return R given a state-action pair, and $y = r + \gamma \max_{a'} Q(s', a')$ is the target value estimated using the optimal Bellman Equation. Note that by assuming $P(R|s, a) = \mathcal{N}(R; \mu, \sigma)$ where $\mu = Q(s, a)$ and σ a constant, then maximizing the likelihood $P(R = y|s, a)$ is equivalent to the Temporal Difference learning objective [16, 22, 32]:

$$\max_Q \log P(R = y|s, a) = \max_Q \log \left[\alpha \exp \left(-\beta \left(Q(s, a) - y \right)^2 \right) \right]$$
$$\Leftrightarrow \min_Q \left(Q(s, a) - y \right)^2 \quad (2)$$

where σ leads to the constants $\alpha, \beta > 0$.

We utilize the variational EM learning strategy [3] to maximize the objective. Denoting the sampled experiences as x, we introduce a variational distribution $q(z|x)$ to approximate the intractable posterior $p(z|x)$, in which we alternate between optimizing for $q(z|x)$ and for other parameterized functions Q^π, Q^ξ, π, ξ. We refer to $q(z|x)$ as the task encoder in our model.
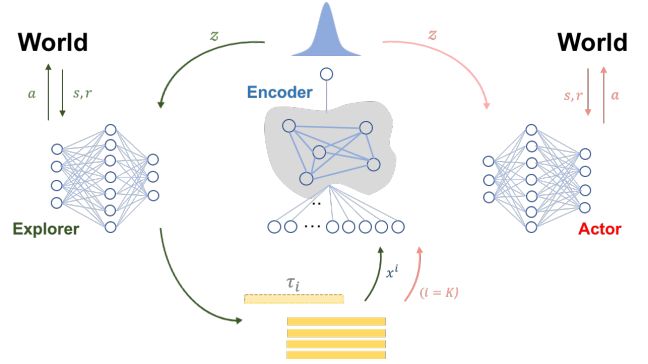

Figure 1: Context-aware task reasoning for RL adaptation. We separate the task into task-exploration, task inference and task-fulfillment. The explorer interacts with the environment to collect experiences for the task encoder to update the belif of task. After an iterative process of $K - 1$ rounds, the task encoder takes all the collections and gives the final task hypothesis to adapt the actor.

For E-step, we maximize a variational lower bound derived from (1) to find the optimal $q(z|x)$. In order to decouple the action and exploration policy, we also introduce an auxiliary experience distribution $p(x)$ and minimize the following free energy:

$$\underset{\substack{p(x), \beta(s,a,s',r) \\ z \sim q(z|x)}}{\mathbb{E}} \left[\left(Q^\pi(s, a, z) - y^\pi \right)^2 + \text{KL}(q(z|x) \| p(z)) \right] + c \quad (3)$$

where in the first term the TD Error derives from $\log \int_Z P(R = y^\pi | s, a, z) p(z) dz$, and the constant term c comes from the second term in (1) which is irrelevant to z. For the auxiliary distribution $p(x)$, we adopt the exploration policy's sample distribution $\rho(\cdot)$, as ξ aims to sample task-informative experiences for task inference. Hence the E-step objective $\mathcal{J}_e(q)$ has the following form:

$$\underset{\substack{x \in \mathcal{X}, z \sim q(z|x) \\ (s,a,s',r) \in \mathcal{B}}}{\mathbb{E}} \left[\left(Q^\pi(s, a, z) - y^\pi \right)^2 + \text{KL}(q(z|x) \| p(z)) \right] \quad (4)$$

For M-step, given $q(z|x)$, we minimize the following free-energy objective $\mathcal{J}_m(\pi, \xi, Q^\pi, Q^\xi)$ derived from (1) :

$$\underset{\substack{x \in \mathcal{X} \\ z \sim q(z|x) \\ (s,a,s') \sim \mathcal{B}}}{\mathbb{E}} \left[\left(Q^\pi(s, a, z) - y^\pi \right)^2 \right] + \underset{\substack{x \in \mathcal{X} \\ z \sim q(z|x) \\ (s,a,s') \sim \mathcal{X}}}{\mathbb{E}} \left[\left(Q^\xi(s, a, z) - y^\xi \right)^2 \right]$$
$$(5)$$

Based on this variational EM algorithm, our approach learns dual agents for task-exploration and task-fulfillment respectively. The meta-training process interleaves the data collection process with the alternating EM optimization process, as shown in Alg. 1.

In contrast to [14, 25], our variational EM formulation derives from a unified learning objective and enables fast learning of dual agents with different roles. The explorer learns the ability of **task-exploration** that targets at efficient exploration to (actively) collect sufficient task information for task inference, while the actor learns for **task-fulfillment** that accomplishes the task towards high rewards. Both the explorer and the actor are task-conditioned, so that they are able to perform temporally-coherent exploration for different goals, given the current task hypothesis [10].

input : Meta-train tasks $T = \{\mathcal{T}_i\}_{1:M}, \sim p(\mathcal{T})$, training
steps N, E-step e, M-step m, meta-batch size B,
task sample size C for buffer update, learning
rate λ.

output : $q(z|x), \pi, \xi, Q^\pi, Q^\xi$.

1 Initialize $q, \pi, \xi, Q^\pi, Q^\xi$ with network parameters θ, ϕ,
  buffer $\mathcal{B}$ for π, buffer X for ξ.
2 **for** *each task* $\mathcal{T}_i \in T$ **do**
3   collect episodes into buffers $\mathcal{B}, X$ with the policies
     π, ξ respectively.
4 **end**
5 **while** *not converged* **do**
6   randomly sample C tasks from T to form a set T_C.
7   **for** *task* $\mathcal{T}_i \in T_C$ **do**
8   add episodes into buffers $\mathcal{B}, X$ with the policies.
9   **end**
10   **for** *each step in N training steps* **do**
11   randomly sample B tasks from T to form a set T_B.
12   **if** *step* $\mod (e + m) < e$ **then**
13   get the inputs (E-step):
14        $\{x_q\}_{1:B} \sim X, \{(s,a,s',r)_\pi\}_{1:B} \sim \mathcal{B}$.
15   compute the loss: $\mathcal{L}(q;\theta) \leftarrow \mathcal{J}_e(q)$.
16   update the model: $\theta \leftarrow \theta - \lambda\nabla_\theta\mathcal{L}(q;\theta)$
17   **else**
18   get the inputs (M-step): $\{x_q\}_{1:B} \sim X$,
         $\{(s,a,s',r)_\pi\}_{1:B} \sim \mathcal{B}, \{(s,a,s',r)_\pi\}_{1:B} \sim X$.
19   compute the loss:
         $\mathcal{L}(\pi, \xi, Q^\pi, Q^\xi; \phi) \leftarrow \mathcal{J}_m(\pi, \xi, Q^\pi, Q^\xi)$.
20   update the model:
         $\phi \leftarrow \phi - \lambda\nabla_\phi\mathcal{L}(\pi, \xi, Q^\pi, Q^\xi; \phi)$
21   **end**
22   **end**
23 **end**

Algorithm 1: The Meta-training algorithm

With such a formulation, we need to answer the following questions to achieve efficient and effective task inference to solve the meta-RL problem:

- How to guide the explorer towards active exploration for sufficient task information that is crucial in task inference with few experiences?
- How to achieve effective task inference that captures the relationships between experiences and reduce task uncertainty?
- Since we incorporate off-policy learning algorithms, how to mitigate the train-test mismatch issue due to the use of replay buffer v.s. online rollouts during testing?

We will address the above three questions in Sec. 4, which introduces our task inference strategy in detail.

Meta-test. At the meta-test time, we apply the same data collection procedure as in the meta-training stage. For each task, the explorer $\xi(a|s, z)$ first samples a hypothesis z from the posterior (initialized as $\mathcal{N}(0, 1)$), and then explores optimally according to the hypothesis. The collected experiences x during the exploration are

used by the task encoder $q(z|x)$ to update the posterior of z. This process iterates until the explorer uses up the maximum number of rollouts. All the experiences are fed to the task encoder to produce a final posterior representing the belief of MDPs. We then sample from the posterior to generate a task-conditioned action policy $\pi(a|s, z)$, which interacts with the environment in an attempt to fulfill the task. This process is shown in Fig. 1.

4 TASK INFERENCE STRATEGY

We now introduce our task reasoning strategy for meta-RL with limited data. Our goal is to use the limited exploration to generate experiences with sufficient information for task inference and to effectively infer the task posterior from the given task experiences. In Sec. 4.1, we explain how we learn an explorer that pursues task-informative experiences with randomized behavior and mitigates train-test mismatch. In Sec. 4.2, we elaborate on the network design of the task encoder, which enables relational modeling between task experiences.

4.1 Learning the Exploration Policy

Our exploration policy aims to enrich the task-related information of experiences within limited episodes. To this end, below we introduce two reward shaping methods: first, we increase the coverage of task experiences to obtain task-informative experiences; second, we improve the quality of each sample to have as much information gain as possible.

4.1.1 Increasing Coverage. The first idea of improving exploration is to increase its coverage of task experiences by adding stochasity into the agent's behavior, which is motivated by the empirical observations of the performance obtain with replay buffer (see the off-policy curve in Fig. 9). Off-policy data have more coverage of experiences as the samples are uniformly drawn from multiple different trajectories, compared to samples in online rollouts with smaller variations.

To this end, we encourage the randomized behavior of the explorer by leveraging SAC [11]. SAC derives from the entropy-regularized RL objective [18], which essentially adds entropy to the reward (and value) functions. Note that we optimize both π and ξ with SAC, but for different purposes. SAC favors exploration in policy learning for the actor π, which will run deterministically at meta-test time. In contrast, here SAC is used to guide the explorer towards randomized task-exploration, which is a stochastic policy during deployment.

It is worth noting that training an exploration policy with randomized behavior also helps mitigate train-test mismatch, which is caused by the different data distributions for policy adaptation (buffer data v.s. online rollouts). Our task-exploration explicitly learns the ability for randomized exploration that (empirically) brings the online rollout data distribution closer to the off-policy buffer data.

4.1.2 Improving Quality. To improve the quality of exploration, we also design a reward shaping that guides the explorer towards highly informative experiences. However, it is non-trivial to quantify the informativeness in a sample. In our case, the explorer collects experiences given a task hypothesis, so we are more interested

in evaluating the mutual information as the reward for the explorer,

$$\tilde{r}_i = \mathcal{I}(z|\tilde{x}; x_i) = \mathcal{H}(q(z|\tilde{x})) - \mathcal{H}(q(\hat{z}|\tilde{x}, x_i)) \quad (6)$$

where we differentiate the new task hypothesis $\hat{z}$ against previously hypothesized task $z \sim q(z|\tilde{x})$, $\tilde{x} = \{x_j\}_{1:K}$ denotes the repository of collected experiences and $x_i \notin \tilde{x}$ is the sample for assessment. Intuitively, we expect the credit $\tilde{r}_i$ to reflect the information gain from the sample x_i given the current hypothesis of task $z \sim q(z|\tilde{x})$.

However, directly computing the mutual information is impractical, as it involves evaluating the new posterior after incorporating each new sample x_i. Instead, we adopt the following proxy for the mutual information:

$$\mathcal{I}(z|\tilde{x}; x_i) \propto \log \frac{1}{p(\hat{z} = z | x_i, \tilde{x})} \quad (7)$$

This proxy implies that we believe a sample x_i brings larger information gain if the new hypothesis $\hat{z}$ is less likely to be the same as the prior belief z. To compute $p(\hat{z} = z | x_i, \tilde{x})$, we apply the Bayes Rule:

$$p(\hat{z} = z | x_i, \tilde{x}) = \frac{p(R = y(\tilde{x}), \hat{z} = z | x_i, \tilde{x})}{P(R = y(\tilde{x}) | \hat{z} = z, x_i, \tilde{x})} \quad (8)$$

As the joint distribution of $(\hat{z}, R)$ is constant given $(x_i, \tilde{x})$, the posterior of $\hat{z}$ is proportional to the inverse of the likelihood. Similar to Sec. 3, we use the state-action value function to compute the likelihood $P(R = y(\tilde{x}) | x_i, \tilde{x})$, but we instead assume a laplace distribution as we empirically find it is more stable to use the L1-norm than the L2-norm induced by a Gaussian. As a result, we have the following score function that gives the shaped reward $\tilde{r}_i$:

$$\tilde{r}_i = \lambda \log P(R = y(\tilde{x}) | \hat{z} = z, x_i, \tilde{x}) \quad (9)$$

$$= \lambda \left\| Q(s_i, a_i, z) - \left(r_i + \gamma \max_{a_i'} Q(s_i', a_i') \right) \right\|_{L_1} \quad (10)$$

where the hyperparameter λ is the reward scale, and the greedy policy $\pi(a|s)$ is learned to compute $\max_a Q(s, a)$ [11, 20, 29].

4.2 Context-aware Task Encoder

Our task inference network computes the posterior of latent variable z given a set of experience data, aiming to extract task information from experiences. To this end, we design a network module with the following properties:

(1) *Permutation-invariant*, as the output should not vary with the order of the inputs.
(2) *Input size-agnostic*, as the network would encounter variable size of inputs within the arbitrary number of rollouts.
(3) *Context-aware*, as extracting cues from a single sample should incorporate the context formed by other samples[1].

Specifically, we adopt a latent Graph Neural Network architecture [43], which integrates self-attention with learned weighted-sum aggregation layers. Formally, we introduce a set of latent node features $H = [h_1, \cdots, h_d]^T$ where $h_i \in \mathbb{R}^c$ is a c-channel feature vector same as the set of input node features $X = [x_1, \cdots, x_n]^T$. Note that n can be arbitrary number while d should be a fixed

[1]Imagine in a 2d-navigation task where the agent aims to navigate to a goal location, a sample may indicate the possible location of the goal due to the high rewards, and can further eliminate possibilities by another sample that shows what locations are not possible by its low rewards.
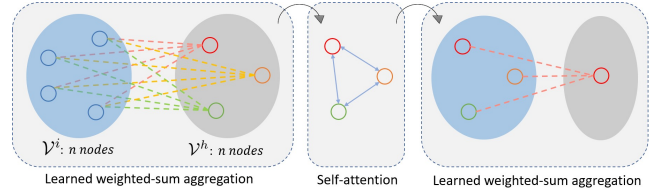

Figure 2: The task encoder. The first aggregation constructs a bipartite graph with full connections from the n nodes in V^i to the c latent nodes in V^h. Self-attention operates on V^h, which are assembled to one latent node in the second aggregation.

hyperparameter. We construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the $n + d$ nodes and full connections from all of the input nodes to each of the latent nodes, i.e., $(v_i, v_j) \in \mathcal{E}, v_i \in \mathcal{V}^i, v_j \in \mathcal{V}^h$ where $\mathcal{V}^i, \mathcal{V}^h$ refers to the set of input nodes and latent nodes respectively. The graph network module is illustrated in Fig. 2.

The output of the aggregation layer f_{AGG} is the latent node features H, which are computed as follows:

$$h_k = \sum_{j=1}^{n} \psi(x_j, \phi_k) x_j, 1 \le k \le d$$

where $\psi(x, \phi_k)$ is a learned affinity function parameterized by ϕ_k that encodes the affinity between any input node x and the kth latent node. In practice, we instantiate this function as the dot product followed with normalization, i.e., $\text{softmax}_j(\phi_k x_j^T)$.

We then combine the above aggregation layer with the following self-attention layer f_{ATN}:

$$\tilde{x}_j = \sum_{i=1}^{n} f(x_i, x_j) x_j, 1 \le i \le n$$

where we use the the scaled dot product attention [34].

Following [43], we propagate messages through a shared space with full-connections between latent nodes. We first pass the input nodes through an aggregation layer with d latent nodes, where $d << n$, then perform self-attention on the latent nodes (for multiple iterations), and finally pass the latent nodes through another aggregation layer with 1 final latent node to obtain the final output, i.e., $f_{\text{AGG}} \circ f_{\text{ATN}} \circ f_{\text{AGG}} : \mathbb{R}^{n \times c} \mapsto \mathbb{R}^{d \times c} \mapsto \mathbb{R}^{d \times c} \mapsto \mathbb{R}^{1 \times c}$. The final output provides the parameters for $q(z|x)$ which is a Gaussian distribution. The network can be viewed as a multi-stage summarization process, in which we group the inputs into several summaries, and operate on these summaries to compute the relationships between entities, and produce a final summary of the entities and relationships.

5 EXPERIMENTS

In this section, we demonstrate the efficacy of our design and the behavior of our method, termed CASTER (shorthand for *Context-Aware task encoder with Self-supervised Task ExploRation for efficient meta-reinforcement learning*), through a series of experiments. We first introduce our experimental setup in Sec. 5.1. In Sec. 5.2, we evaluate our CASTER against three meta-RL algorithms in terms of sample efficiency. We then compare the behavior of CASTER with PEARL regarding overfitting and exploration in Sec. 5.3. Finally, in
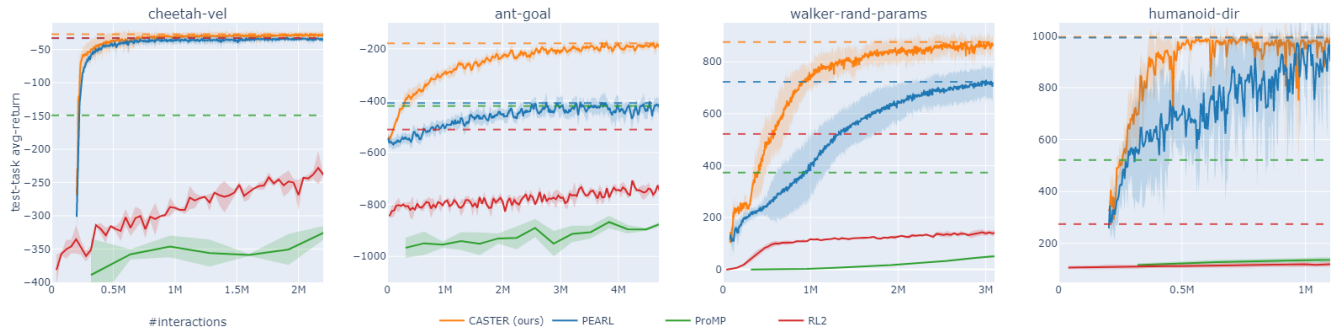
Figure 3: Training efficiency. The test-task performance versus the number of interactions with the environment during meta-training. The dash lines represent the asymptotic performance of each method.

Figure 4: Testing efficiency. The x-axis denotes the number of trajectories used as adaptation data.

Sec. 5.4, we conduct ablation study on our design of the exploration strategy and the task encoder.

5.1 Experiemtal Setup

We evaluate our method on four benchmarks proposed in [26][2] and an environment introduced in [25] for ablation. They are implemented on the OpenAI Gym [4] with the MuJoCo simulator [33]. All of the experiments characterize locomotion tasks, which may vary either in the reward function or the transition function. We briefly describe the environments as follows.

- **Half-cheetah-velocity.** Each task requires the agent to reach a different target velocity.
- **Ant-goal.** Each task requires the agent to reach a goal location on a 2D plane .
- **Humanoid-direction.** Each task requires the agent to keep high velocity without falling off in a specified direction.
- **Walker-random-params.** Each task requires the agent to keep high velocity without falling off in different system configurations.
- **Point-robot.** Each task requires a point-mass robot to navigate to a different goal location on a 2D plane.

We adopt the following evaluation protocol throughout this section: first, the estimated per-episode performance on each task is averaged over (at least) three trials with different random seeds; second, the test-task performance is evaluated on the held-out meta-test tasks and is an average of the estimated per-episode performance over all tasks with at least three trials. More details about the experiments can be found at our Github repository.

5.2 Performance

In this section, we compare with three meta-RL algorithms that are the representatives of the three lines of works mentioned in Sec. 2.1: 1) inference-based method, PEARL [25]; 2) optimization-based method, ProMP [26]; 3) black-box method, RL^2 [7].

For those baselines, we reproduce the experimental results via their officially released code, following their proposed meta-training and testing pipelines. We note that prior works [7, 26] are not designed to optimize for sample efficiency but we keep their default hyperparameter settings in order to reproduce their results.

To demonstrate the training efficiency and testing efficiency, we plot the test-task performance as a function of the number of samples. Fig. 3 shows the comparison results on the training efficiency. Here the x-axis indicates the number of interactions with the environment used to collect buffer data. At each x-tick, we evaluate the test-task performance with **2** episodes for all methods. While for testing efficiency, the x-axis refers to the number of

---

[2]Two experiments "cheetah-forward-backward" and "ant-forward-backward" are not used because they only include two tasks (the goal of going forward and backward), and do not match the meta-learning assumption that there is a task distribution from which we sample the meta-train task set and a held-out meta-test task set. Such benchmark does not provide convincing evidence for the efficacy of meta-learning algorithms.
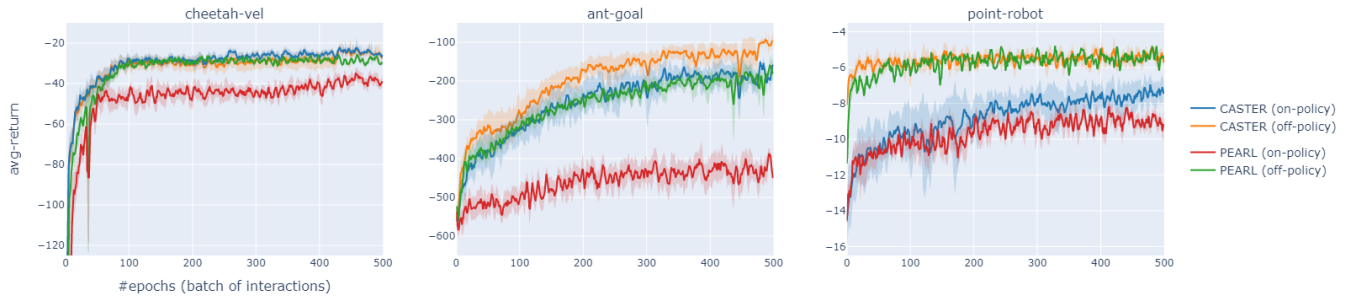
Figure 5: Overfitting in off-policy meta-RL. Each column in the plot corresponds to a different environment. We pick three environments most prone to meta-overfitting, i.e., 'cheetah-vel', 'ant-goal' and 'point-robot' from left to right.

adaptation episodes used by different methods to perform policy adaptation (or task inference).

We can see CASTER outperforms other methods by a sizable margin: CASTER achieves the same status of performance with much fewer environmental interactions (up to 400%) while being able to reach higher performance (up to 300%). PEARL and CASTER incorporate off-policy learning and naturally enjoy an advantage in training sample efficiency. Our CASTER achieves better performances due to two reasons. On the one hand, efficient exploration potentially offers CASTER richer information within limited episodes (two in Fig. 3), and pushes higher the upper bound of accurate task inference. On the other hand, context-aware relational reasoning extracts information effectively from the task experiences, and thus improves the ability of task inference.

Testing efficiency is shown in Fig. 4, and CASTER still stands out against other methods. Notably, CASTER achieves large performance boost with the first several episodes, indicating that the learned exploration policy is critical in task inference. By contrast, other models either fail to improve from zero-shot performance (e.g. flat lines of RL^2, PEARL in *humanoid-dir*, ProMP in *walker-rand-params*) via exploration or have unstable performance that drops after improvement (e.g., PEARL in *cheetah-vel* and *ant-goal*).

5.3 Understanding CASTER's Behavior

5.3.1 Meta-overfitting. As mentioned in 4.1, meta-overfitting arises due to train-test mismatch, an issue particular for meta-RL methods that incorporate off-policy learning. We compare the test-task performance obtained with different adaptation data distribution, i.e., the off-policy buffer data v.s. the on-policy exploration data, to see the performance gap when the data distribution shifts. We pick three environments that we find most prone to train-test mismatch, i.e., *cheetah-vel*, *ant-goal*, *point-robot*. Note that in the experiments, the number of transitions in the off-policy data equals the number of transitions in two episodes of the on-policy data.

Fig. 5 shows the results on the three environments^3. CASTER takes a large leap towards bridging the gap between train and test sample distribution, reducing up to 75% performance drop as compared to PEARL. This can be credited to better stochasity inherent in the explorer's episodes (Sec. 5.3.2) and better task inference of CASTER. The stochasity in the online rollouts brings its distribution

---

^{3}The *ant-goal* curve reproduced with PEARL's public repo has a discrepancy from the result reported in [25], which we suspect to be the train-task performance with off-policy data. We recommend the reader to check with the publicly available code.
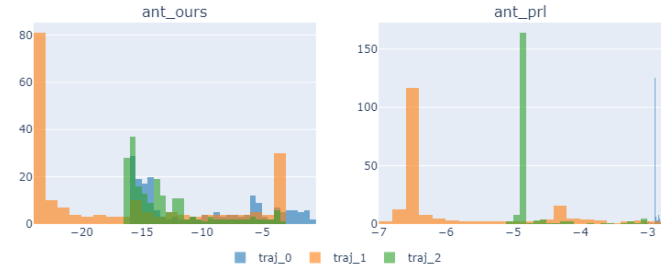

Figure 6: Learned exploration behavior in *ant-goal*. The reward histogram consists of 3 consecutive trajectories. Left: CASTER and Right: PEARL. Different color denotes different trajectories.

closer to the off-policy data distribution, since at meta-train time data is randomly sampled from the buffer. Task inference that is aware of relations between samples better extracts the information in the few trajectories, which also narrow the gap by improving task inference.

5.3.2 The Learned Exploration Strategy. We investigate the exploration process by visualizing the histogram of the rewards in the collected trajectories. We take three consecutive trajectories rolled out by the explorer on two benchmarks : 1) the reward-varying environment *ant-goal*, 2) the transition-varying environment *walker-rand-params*. We compare with PEARL since it also resorts to posterior sampling for exploration [23, 24, 31].

In Fig. 6, it is shown that CASTER's exploration spans a large reward region of [−25, 0], while PEARL ranges from −7 to −3. This is consistent with our goal to increase the coverage of task experiences within few episodes, which potentially increases the chances to reach the goal and enables the task encoder to reason with both high rewards and low rewards, i.e., what region of state space might be the goal and might not. By contrast, PEARL tends to exploit the higher extreme values at the risk of following a wrong direction (e.g., in the second plot, PEARL proceeds to explore in a narrow region of relatively lower reward than the preceding episode).

In Fig. 7, the two methods are evaluated in the transition-varying environment *ant-goal* in which higher rewards do not necessarily carry the information of the system parameters. In this case, PEARL clings to higher reward regions as expected, while CASTER favors lower reward region. Supported by the results in Fig. 3,4, PEARL's exploration is sub-optimal. CASTER performs better because the
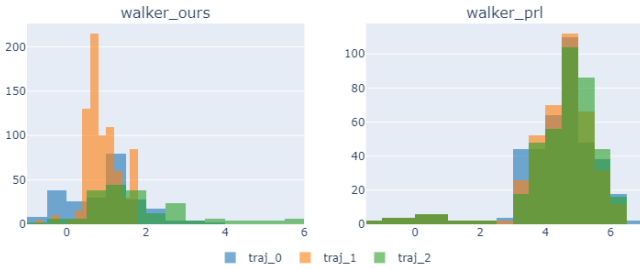
Figure 7: Learned exploration behavior in *walker-rand-params*.

Figure 8: Ablation study of the task encoder. We show the test-task performance with on-policy data (left) and off-policy data (right).

exploration is powered by information gain, and the explorer discovers task-discriminative experiences that happen to be around lower reward region.

5.4 Ablations

In this section, we investigate our design choices of the exploration strategy and the task encoder via a set of ablative experiments on the *point-robot* environment. We also provide the test-task performance with off-policy data (adaptation data from a buffer collected beforehand) to eliminate the effect of insufficient data source for task inference that induces train-test mismatch.

5.4.1 The Task Encoder. PEARL [25] builds its task encoder by stacking a Gaussian product (Gp) aggregator on top of an MLP, while we propose to combine self-attention with a learned weighted-sum aggregator for better relational modeling. We examine the following models: 1) *Enc(Gp)-Exp(None)* that uses the Gp task encoder, 3) *Enc(WS)-Exp(None)* that uses a learned weighted-sum aggregator without self-attention for graph message-passing, 3) *Enc(GNN)-Exp(None)* that uses the proposed GNN encoder, 4) and *Enc(GNN)-Exp(RS)* the proposed model. Note that for the first three baselines, the explorer is disabled to eliminate the impact of the exploration policy.

In Fig. 8, all models tend to converge to similar asymptotic performance with off-policy data. However *Enc(WS)-Exp(None)* learns much slower than its counterparts, while *Enc(Gp)-Exp(None)* seems a reasonable design w.r.t. the off-policy performance. We conjecture that it can be hard for *Enc(Gp)-Exp(None)* to learn a weighting strategy as Gp, in which the weights are determined by the relative importance of a sample w.r.t. the whole pool of samples ($\sigma_i^2 / \sum \sigma_j^2$), and *Enc(Gp)-Exp(None)* has no access to other samples when computing the weight of each sample. By contrast, *Enc(GNN)-Exp(None)*
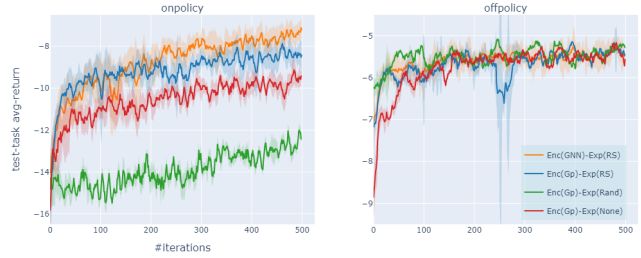

Figure 9: Ablation study of exploration. We show the test-task performance with on-policy data (left) and off-policy data (right).

provides more flexibility for the final weighted average pooling, by incorporating the interactions between samples. Such relational modeling enables it to extract more task statistics within the few episodes, superior to Gp in the on-policy performance.

5.4.2 The Exploration Strategy. We aim to show the efficacy of the proposed reward shaping for task-exploration. The baseline models are: 1) *Enc(Gp)-Exp(None)* that uses no exploration policy, 2) *Enc(Gp)-Exp(Rand)* that uses a (uniformly) random explorer, 3) *Enc(Gp)-Exp(RS)* that uses an explorer guided by the proposed reward shaping. Here the Gp task encoder is used by default.

As shown in Fig. 9, all models perform equally in terms of asymptotic performance with off-policy data, since sufficient coverage of experiences are inherent in the data randomly sampled from buffers. This demonstrates the significance of improving coverage of task experiences for task inference.

For the on-policy performance, we can see *Enc(Gp)-Exp(Rand)* suffers from significant overfitting, as the coverage brought by the randomness doesn't suffice to benefit it within few episodes. *Enc(Gp)-Exp(RS)* performs much better than *Enc(Gp)-Exp(Rand)* because high rewards is an informative guidance in a reward-varying environment. Our approach combines the merits of both broad coverage over task experiences and informative guidance regarding task relevance, and hence achieves the best performance.

6 CONCLUSION

We divide the problem of meta-RL into three sub-tasks, and take on probabilistic inference via variational EM learning. We thus present CASTER, a novel meta-RL method that learns dual agents with a task encoder for task inference. CASTER performs efficient task-exploration via a curiosity-driven exploration policy, from which the collected experiences are exploited by a context-aware task encoder. The encoder is equipped with the capacity for relational reasoning, with which the action policy adapts to complete the current task. Through extensive experiments, we show the superiority of CASTER over prior methods in sample efficiency, and empirically reveal that the learned exploration strategy efficiently acquires task-informative experiences with randomized behavior, which effectively helps mitigate meta-overfitting.

7 ACKNOWLEDGEMENT

REFERENCES

[1] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. 2016. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*. 4502–4510.

[2] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

[3] Christopher M Bishop. 2006. *Pattern recognition and machine learning.* springer.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[5] Noam Brown and Tuomas Sandholm. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* 359, 6374 (2018), 418–424.

[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[7] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779* (2016).

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1126–1135.

[9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1263–1272.

[10] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. 2018. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*. 5302–5311.

[11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290* (2018).

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[13] Yedid Hoshen. 2017. Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems*. 2701–2711.

[14] Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. 2019. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424* (2019).

[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[16] Vijay R Konda and John N Tsitsiklis. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*. 1008–1014.

[17] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and brain sciences* 40 (2017).

[18] Sergey Levine. 2018. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909* (2018).

[19] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* (2015).

[20] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[21] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2017. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141* (2017).

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[23] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped DQN. In *Advances in neural information processing systems*. 4026–4034.

[24] Ian Osband, Daniel Russo, and Benjamin Van Roy. 2013. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*. 3003–3011.

[25] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. 2019. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *arXiv preprint arXiv:1903.08254* (2019).

[26] Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. 2018. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784* (2018).

[27] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*. 4967–4976.

[28] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144.

[29] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms.

[30] Bradly C Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. 2018. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118* (2018).

[31] Malcolm Strens. 2000. A Bayesian framework for reinforcement learning. In *ICML*, Vol. 2000. 943–950.

[32] Gerald Tesauro. 1995. Temporal difference learning and TD-Gammon. *Commun. ACM* 38, 3 (1995), 58–68.

[33] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[35] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[36] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, et al. 2019. AlphaStar: Mastering the real-time strategy game StarCraft II. *DeepMind Blog* (2019).

[37] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2015. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391* (2015).

[38] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*. 3630–3638.

[39] Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matthew Botvinick. 2016. Learning to reinforcement learn. *CoRR* abs/1611.05763 (2016). arXiv:1611.05763 http://arxiv.org/abs/1611.05763

[40] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2018. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7794–7803.

[41] Tianbing Xu, Qiang Liu, Liang Zhao, and Jian Peng. 2018. Learning to explore via meta-policy gradient. In *International Conference on Machine Learning*. 5459–5468.

[42] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. 2017. Deep sets. In *Advances in neural information processing systems*. 3391–3401.

[43] Songyang Zhang, Shipeng Yan, and Xuming He. 2019. LatentGNN: Learning Efficient Non-local Relations for Visual Recognition. *arXiv preprint arXiv:1905.11634* (2019).