M3: Modularization for Multi-task and Multi-agent Offline Pre-training

Linghui Meng*
Institute of Automation, Chinese Academy of Sciences
School of Artificial Intelligence, University of Chinese Academy of Sciences
Beijing, China
menglinghui2019@ia.ac.cn

Jingqing Ruan*
Institute of Automation, Chinese Academy of Sciences
School of Future Technology, University of Chinese Academy of Sciences
Beijing, China
ruanjingqing2019@ia.ac.cn

Xuantang Xiong
Institute of Automation, Chinese Academy of Sciences
School of Artificial Intelligence, University of Chinese Academy of Sciences
Beijing, China
xiongxuantang2021@ia.ac.cn

Xiyun Li
Institute of Automation, Chinese Academy of Sciences
School of Future Technology, University of Chinese Academy of Sciences
Beijing, China
lixiyun2020@ia.ac.cn

Xi Zhang
Institute of Automation, Chinese Academy of Sciences
Beijing, China
sheryl.zhangxi@gmail.com

Dengpeng Xing†
Institute of Automation, Chinese Academy of Sciences
School of Artificial Intelligence, University of Chinese Academy of Sciences
Beijing, China
dengpeng.xing@ia.ac.cn

Bo Xu†
Institute of Automation, Chinese Academy of Sciences
School of Artificial Intelligence, University of Chinese Academy of Sciences
Beijing, China
xubo@ia.ac.cn

ABSTRACT

Learning a multi-task policy is crucial in multi-agent reinforcement learning (MARL). Recent work has focused on learning in the context of online multi-task reinforcement learning, where a policy is jointly trained from scratch, aiming to generalize well to few-shot or even zero-shot tasks. However, existing online methods require tremendous interactions and are therefore unsuitable for environments where interactions are expensive. In this work, we novelly introduce the modularization for multi-task and multi-agent offline pre-training (M3) to learn high-level transferable policy representations. We claim that the discrete policy representation is critical for multi-task offline learning and accordingly leverage contexts as a task *prompt* to enhance the adaptability of pre-trained models to various tasks. To disentangle multiple agents of variation under heterogeneous and non-stationary properties even though they receive the same task, we employ an agent-invariant VQ-VAE to identify each of the multiple agents. We encapsulate the pre-trained model as part of an online MARL algorithm and fine-tune it to improve generalization. We also theoretically analyze the generalization error of our method. We test the proposed method on the challenging StarCraft Multi-Agent Challenge (SMAC) tasks, and empirical results show that it can achieve supreme performance in few-shot or even zero-shot settings across multiple tasks over state-of-the-art MARL methods.

KEYWORDS

Offline multi-agent reinforcement learning; Multi-task learning; Representation learning

ACM Reference Format:
Linghui Meng*, Jingqing Ruan*, Xuantang Xiong, Xiyun Li, Xi Zhang, Dengpeng Xing†, and Bo Xu†. 2023. M3: Modularization for Multi-task and Multi-agent Offline Pre-training. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 10 pages.

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) is well suited for solving complex decision problems by controlling multiple agents and has been preliminarily applied to various control scenarios [12], including robot systems [33], autonomous driving [59], and resource

* These authors contribute equally to this work. † Corresponding authors.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

utilization [9, 20]. As opposed to single-agent reinforcement learning, the main challenge of MARL is that all the agents must interact and that their returns depend on the overall behavior, which usually requires many samples. After sufficient exploration, agents learn to collaborate effectively, but MARL still induces millions of interactions. Therefore, the training complexity caused by the multi-agent exploration is high, resulting in low sample efficiency. In addition to the high sample efficiency requirement, the trained policy needs to generalize to many tasks to avoid extra training costs [5]. Particularly in the training and inference stages of new scenarios, sample efficiency and generalizability play a crucial role in reinforcement learning. To reduce the complexity of multiple tasks, multi-task reinforcement learning (MTRL) is a promising approach for training agents in the real world [42, 54].

Generally, MTRL, in the single-agent case, trains policies via multi-task learning based on the learning architecture in three ways, which learn shared architectures to learn how to share parameters among tasks: branch sharing, modular sharing, and conditional architecture. 1) By sharing a backbone network and task-specific heads across multiple tasks, agents are trained to learn different policies in parameter-sharing methods [10]. Multi-task learning typically involves hand-crafted networks with shared initial layers that branch out at an arbitrary point or task-specific networks with additional fusion mechanisms and feature sharing. 2) As opposed to branch sharing, those methods provide an explicit representation of the policy through a routing mechanism [50]. In the hidden layer, policies can be routed from different modules to specific tasks. 3) Furthermore, [39] leverages task metadata to learn context-based representations that guide policy-learning experiences.

The above methods can be naturally extended to multi-agent setup for multi-task multi-agent reinforcement learning (MT-MARL) by sharing policies across agents or decentralized multiple agents [30]. MT-MARL is an open and challenging problem. Many practical multi-task scenarios require the coordination of heterogeneous agents, such as UAV cooperation and football games, in which vehicles or players should take different formations and play different roles across tasks for better coordination. In such scenarios, sharing policies across agents may induce homogenization behaviors, and handcrafting tasks for each agent is tedious and sometimes difficult. Therefore, multi-agent policy learning is irreplaceable, and we focus on another branch in which agents receive the same task context but act differently. An easy-to-understand example is a football game where multiple players work together for the same winning goal, but different roles offer different policies. Despite building on the multi-agent relations, they miss out on conducting a multi-task multi-agent policy by leveraging *offline data* and fine-tuning it online. There are considerable offline multi-task trajectories and task descriptions in practical decision-making problems which have not been fully employed. Early research attempts to pad the input and output across different tasks to pre-train a unified multi-task policy offline for the multi-agent problem [26].

This paper investigates multi-task learning for MARL and proposes a novel offline pre-training method, M3. We train the policy via a context-based generative model while quantizing the latent space, then fine-tune it online in a simulated environment. Besides, a novel policy representation learning scheme was proposed for optimization based on theoretical analysis. In the offline pre-training

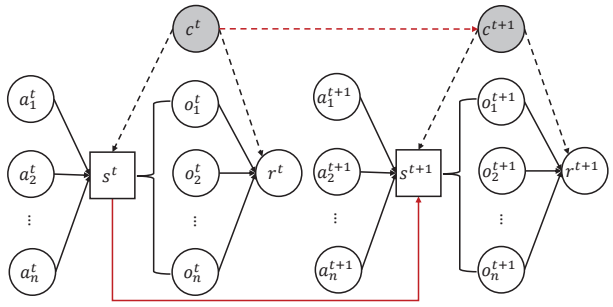


Figure 1: The graphical model of Contextual Markov Games (CMG) ranges from timestep t to $t+1$. The red line means the transition to the next step. POMDP can split states s as local observations $\{o_i\}_{i=1}^n$.

stage, we split the trajectories of multiple agents individually as the input of an encoder and partition the hidden space for discriminating each agent. To reconstruct input trajectories, partitioned agent-sharing representations are combined with task embeddings and then quantized according to a learnable codebook. In addition, the encoded representations are leveraged to map the expected action. In the online fine-tuning stage, the pre-trained encoder is loaded as an actor to construct a PPO-based algorithm for retraining. Multiple agents then learn this policy by interacting with the environment. The effect of each technique in M3, which revolves around (and is necessary for) the multi-agent multi-task setting includes: 1) We use VQ-VAE to extract transferable discrete policy representations in the latent space to deal with *multi-task policy composition* as the modularization method. 2) We embed task context information into the policy representation to prompt the *multi-agent policy under each task*. 3) We design agent-specific shared modules to discriminate *each agent's specific role in each task*. The ablation results demonstrate their importance.

Key contributions of this paper are summarized as follows: 1) We present the first solution specifically designed for the offline MT-MARL problem. 2) We introduce a novel discrete policy representation learning method for multi-task transfer with theoretical analysis, where we propose an agent-invariant network to extract agent-sharing policy representation and an efficient way for context incorporation. 3) Our method achieves state-of-the-art results on challenging MT-MARL tasks.

2 RELATED WORK

Offline MARL. Typically, in offline (MA)RL, a policy is trained based on previously collected static demonstrations or trajectories offline and evaluated directly in the online environment [13, 24]. It is mainly motivated by the expensive cost of interacting with the environment. In addition, the offline RL algorithms will constrain the off-policy learning to fix the distributional shift problem between offline and online stages [14, 22, 51]. In addition to these, pre-training a policy from offline data and fine-tuning it by interacting with the online environment arises in the need for efficient reinforcement learning [27]. Furthermore, Chen et al. [6], Janner et al. [21] propose that utilizing Transformers to encode trajectories offline outperforms many state-of-the-art offline RL methods [45]. In addition, Meng et al. [26] extend the decision transformer to the

multi-agent field, aiming at pre-training a policy and fine-tuning it on downstream tasks. They collect samples from demonstrations in which the buffer of the state-of-the-art policies on SMAC is used as the multi-agent offline datasets [36]. They mainly conduct single-task offline pre-training, and online fine-tuning, the related to us is its variants by padding zero across multiple tasks. Therefore, we set its variants as our baseline in experimental validation. In this work, we apply this paradigm to the multi-task domain. Our approach learns the policy representation explicitly and modularizes the hidden space in pre-training with a multi-task learning process. In addition, we represent the policy well to capture a policy that can be generalized quickly to subtasks.

Multi-task learning in RL. Multi-task reinforcement learning is mainly divided into two directions in research: online and offline multi-task RL [12]. Early attempts at using multi-task in RL online regard a single environment as a task and aggregate multiple environments together [2, 5, 8]. A line for this branch is to share a backbone across multiple tasks with task-specific heads [10, 19], learn to effectively communicate among agents employing meta-reinforcement learning to identify communication behaviors and extract information that facilitates the multi-task training process [44], learn to distillation a unified policy from single-task [30], or learn a skill-based hierarchical framework to generalize [28]. Another line is to set the modular network for multiple tasks using different combinations [11, 50]. Recently, the other multi-task RL method has attracted more attention in offline learning by utilizing static experiences collected from multiple tasks [26] via meta-learning for fine-tuning or conservatively sharing data [53, 57]. Also, related works on multi-objective MARL [34], while mainly focusing on the perspective of utility or reward in a fine granularity that is parallel with the task level. However, it has not been investigated in offline MT-MARL settings. We split the offline datasets based on the minimap types, pre-train a policy offline, and fine-tune it in few-shot or zero-shot settings. A closely related line of work is modular multi-task learning [3, 11, 50], which learns the compositional models representing the different modules of the multiple tasks to generalize to unseen tasks. In addition, Sodhani et al. [38, 39] attempt to represent the policy in the latent space by incorporating the task information for the multi-task problem. In this work, we propose learning discrete policy representations from offline multi-task data. Furthermore, similar to the intuition of role-based methods for assigning agents with various roles automatically [7, 47, 48], we further propose to improve the intra-agent transfer to combine multi-agent with the offline multi-task settings by introducing the agent-invariant module for fine-tuning online.

Representation Learning in RL. The need to learn generalizable representations of observations, dynamics, actions, or policies arises in the RL field. Srinivas et al. [40], Stooke et al. [41], Zhang et al. [56] use contrastive learning and bisimulation methods to extract hidden representations from observations. In action representation, Agarwal et al. [1] attempt to learn transferable behaviors across different scenarios. In addition, Hafner et al. [16, 17], Ozair et al. [32] learn the world model in the latent space for better planning, and Grover et al. [15] learns policy representations in the multi-agent field to transfer the learned policy to another task. However, policy representation learning in offline MT-MARL has not been well explored. Discrete latent-variable models dominate many challenging tasks under semi-supervised and even unsupervised learning, such as speech recognition and image classification [4, 35, 58]. We hypothesize that discrete learnable hidden representations can use VQ-VAE as an inductive bias to assist multi-task learning with different combinations. In this work, we leverage the discrete latent-variable model, VQ-VAE [31], to learn policy representations from offline multi-agent trajectories, in which the latent variable priors should be discretized by quantizing the continuous hidden space in a variational autoencoder.

3 PRELIMINARY

3.1 Contextual Markov Games

We denote the Contextual Markov Games (CMG) as an extension of the Contextual Markov Decision Process (CMDP) proposed by [18] in a multi-agent setting and depict it in Figure 1.

Definition 3.1. ***(Contextual Markov Games):*** *A Contextual Markov Game with n agents can be regarded as an extension of the Contextual Markov Decision Process. Therefore, it is also defined with a tuple* $\langle C, \mathcal{S}, \mathcal{A}, \mathcal{M}, n, \gamma \rangle$. $\mathcal{M}$ *is a function which maps a context* $c \in C$ *to the reward and transition function* $\mathcal{M}(c) = \{R^c, \mathcal{P}^c\}$.

The contexts can be viewed as task prompts in the multi-task setting, where each task has a context for demonstrating the meta-information. Here $c \in C$ denotes the context corresponding to a task. C is the context space, $\mathcal{S}$ is the shared state space, $\mathcal{A}$ is a joint action space across n agents: $\mathcal{A}_1 \times \cdots \times \mathcal{A}_n \rightarrow \mathcal{A}$, one for each agent. $\mathcal{P}^c(s^{'}|s, a_1, \ldots, a_n) : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ represents the transition probability, $R^c(s, a_1, \ldots, a_n) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ denotes the reward function, γ is the discount factor, and n is the agent number. In this paper, we consider a shared state space but partial access to the global state with local observation $o_i \in \Omega$ according to an observation function $O(s, i), i \in [1, n]$ for each agent, which is also called Partial-Observed MDP (POMDP) [29]. We consider the CMG, where each agent $i \in [1, n]$ maps the input observation $o_i \in \Omega$ and context $c \in C$ for a specific task to the action with the policy $\pi_i(a|o_i, c) : \Omega \times C \rightarrow \mathcal{A}_i$. Each agent aims to cooperate to maximize their shared long-term team reward $\sum_t \gamma^t r_t$, where γ denotes the discount factor, $r_t \in R^c$ denotes the team reward sharing by n agents at step t. To clarify the notation, we denote the time t for variables of all agents by sub-scripts but the time for variables with agent index by super-scripts. Moreover, multiple agents share a reward that suits the cooperative MARL, especially when applied to the challenging task, the Starcraft multi-agent challenge (SMAC).

3.2 Offline Pre-Training MT-MARL

Multi-Task Offline Pre-Training RL is to learn a policy from offline datasets collected from different tasks and generalize it properly to downstream tasks based on pre-training and fine-tuning. An extension of the above setting in multi-agent is offline pre-training MT-MARL. The goal of offline pre-training MT-MARL in this work is to find a multi-agent policy learned from multi-task datasets that can generalize well on each task in the few-shot or even zero-shot setting following a CMG. Each task presents a different reward function R^c, but we assume that the dynamics, $\mathcal{P}$ are shared across tasks. Under this assumption, we give a solid theoretical analysis of the generalization bound. In spite of this setting not being fully

general, there is a range of practical problem settings in which only the reward changes, including a variety of navigation tasks, distinct manipulation objectives, and a wide range of user preferences. We will explore the relaxed assumption further in the future. In this work, we focus on learning a multi-agent task-conditioned policy $\pi(a_i|o_i, c)$ by sharing parameters. We formulate the problem of policy optimization in terms of finding a policy that maximizes expected return across all tasks:

$$\pi^*(a|o,\cdot) = \arg\max_{\pi} \mathbb{E}_{c\sim C,\pi(\cdot|\cdot,c)}\left[\sum_t \gamma^t R^c(s_t, a_t)\right] \tag{1}$$

Conventional offline MARL in POMDP is concerned with learning policies π using only a given static dataset of transitions $\mathcal{D} = \{(s, o_i, a_i, o_i^{'}, r)\}_{i=1}^{n}$, collected by a behavior policy $\pi_\beta(a_i|o_i)$, without any additional environment interaction to test the performance. In the multi-task setting, the dataset $\mathcal{D}$ is partitioned into multiple subsets in terms of tasks, $\mathcal{D} = \cup_c \mathcal{D}_c$, where $\mathcal{D}_c$ consists of experience under the context c. While algorithms can choose to directly test on the environment with pre-training on task c only on $\mathcal{D}_c$, in this paper, we are interested in generalizing the pre-trained policy online to interact with another environment with context c^{test} is equal to c^{train} or not with the reward function r_c, and learn $\pi(\cdot|o, c)$ on the combined data. The critical challenges of this problem are mainly reflected in reducing the generalization error of learning and transferring in our setting $\Phi = \left|\frac{1}{N}\sum_{j=1}^{N} R(\boldsymbol{\tau}^j, c^{train}) - \mathbb{E}_{\boldsymbol{\tau}\sim\mathcal{D}_{\hat{\pi}}^{'}} R(\boldsymbol{\tau}, c^{test})\right|$. The challenges are caused by a variety of tasks $c^{train} \neq c^{test}$ and data collection sources $\mathcal{D}_\pi \neq \mathcal{D}_{\hat{\pi}}^{'}$, where π denotes the source policy underlying on the offline data, and $\hat{\pi}$ denotes the adapted online learning policy. This paper aims to improve the multi-agent policy learning generalizable representations by combining multiple tasks with offline data.

4 METHODOLOGY

In this section, we first introduce the objective of our M3 for solving the offline pre-training MT-MARL problem defined in Section 3. Then to demonstrate each module in this method, we describe the training paradigm. The overall pipeline of M3 is shown in Figure 2.

4.1 Main Objective

We separately introduce objectives for offline multi-task pre-training and online single-task fine-tuning. In the offline stage, each agent's policy takes the previously collected trajectories as input and optimizes the objective in Equation 6. In the next stage, the pre-trained policy is loaded as an initial policy for online fine-tuning and interacts with the environment to maximize the expected return in Equation 7. We provide the pseudocode of our algorithm M3 for offline pre-training and online fine-tuning.

Offline Learning. In this stage, given the observation o_i^t and previous action a_i^{t-1} of an agent i at timestep t, we first aim to learn an encoder taking as input the action-observation history, $\boldsymbol{\tau}_i^t$, where $\boldsymbol{\tau}_i^t := \{o_i^j, a_i^{j-1}\}_{j=1}^t$ denotes the set of observations and actions, to represent the trajectories in the invariant hidden space. Then the decoder can reconstruct them with a variational autoencoder by maximizing the $\log p(\boldsymbol{\tau}_i)$. Note that we share the encoder across all agents and tasks. The encoder $p(\boldsymbol{z}|\boldsymbol{\tau}_i)$ encodes the trajectory into the embedding space. In order to transfer intra-agent, we induce the agent-invariant term by partitioning the latent space encoded above. We suppose the hidden space is split into two parts: agent-sharing $\boldsymbol{z_\tau}$ and agent-specific $\boldsymbol{z}_i$ from $\boldsymbol{z}$. Agent-sharing can be leveraged for policy representation across agents. Following CMG, we incorporate the context c, which is the task meta-information from the context space C across the multi-agent tasks. To utilize the task information for each agent in the encoding results with context c, the sharing vector $\boldsymbol{z_\tau}$ is then incorporated with the context information as $\boldsymbol{z}_e(\boldsymbol{\tau}) = p(c)\boldsymbol{z_\tau}$. The encoder output $\boldsymbol{z}_e(\boldsymbol{\tau})$ is then quantized with a learnable expert dictionary to discrete space. In addition, $\boldsymbol{z}_e(\boldsymbol{\tau})$ is also encoded to map the action space to predict the action with $\hat{a}_i^t$. We regard this model as VAE where we can bound $\log p(\boldsymbol{\tau})$ with the ELBO by minimizing the Kullback-Leibler divergence between true priors $q_{\boldsymbol{\tau}}(\boldsymbol{z})$, $q_i(\boldsymbol{z})$ and hidden estimator $p(\boldsymbol{z}_i|\boldsymbol{\tau})$, $p(\boldsymbol{z_\tau}|\boldsymbol{\tau})$ as follows:

$$\begin{aligned} &2\log p(\boldsymbol{\tau}) - \mathbb{E}_{\substack{\boldsymbol{z}_i\sim q_i(\boldsymbol{z}),\\ \boldsymbol{z_\tau}\sim q_{\boldsymbol{\tau}}(\boldsymbol{z})}}[\log p(\boldsymbol{\tau}|\boldsymbol{z}_i, \boldsymbol{z_\tau})] \\ &+ KL[q_i(\boldsymbol{z})||p(\boldsymbol{z}_i)] + KL[q_{\boldsymbol{\tau}}(\boldsymbol{z})||p(\boldsymbol{z_\tau})] \end{aligned} \tag{2}$$

Regarding the maximizing likelihood method, we get the likelihood of trajectories in Equation 3. When the hidden space is partitioned into two parts representing agent-sharing $\boldsymbol{z_\tau}$ and agent-specific $\boldsymbol{z}_i$, the reconstruction loss can be changed. Therefore, we derive the reconstruction objective when inducing the agent-invariant part with ELBO bounds by minimizing the Kullback-Leibler divergence between true priors $q_{\boldsymbol{\tau}}(\boldsymbol{z})$, $q_i(\boldsymbol{z})$ and hidden estimator $p(\boldsymbol{z}_i|\boldsymbol{\tau})$, $p(\boldsymbol{z_\tau}|\boldsymbol{\tau})$ as follows:

$$\begin{aligned} &KL[q_{\boldsymbol{\tau}}(\boldsymbol{z})||p(\boldsymbol{z_\tau}|\boldsymbol{\tau})] + KL[q_i(\boldsymbol{z})||p(\boldsymbol{z}_i|\boldsymbol{\tau})] \\ &= \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\tau}}(\boldsymbol{z})}[\log q_{\boldsymbol{\tau}}(\boldsymbol{z}) - \log p(\boldsymbol{z_\tau}|\boldsymbol{\tau})] \\ &+ \mathbb{E}_{\boldsymbol{z}\sim q_i(\boldsymbol{z})}[\log q_i(\boldsymbol{z}|\boldsymbol{\tau}) - \log p(\boldsymbol{z}_i|\boldsymbol{\tau})] \\ &= 2\log p(\boldsymbol{\tau}) - \mathbb{E}_{\substack{\boldsymbol{z}_i\sim q_i(\boldsymbol{z}),\\ \boldsymbol{z_\tau}\sim q_{\boldsymbol{\tau}}(\boldsymbol{z})}}[\log p(\boldsymbol{\tau}|\boldsymbol{z}_i, \boldsymbol{z_\tau})] \\ &+ KL[q_i(\boldsymbol{z})||p(\boldsymbol{z}_i)] + KL[q_{\boldsymbol{\tau}}(\boldsymbol{z})||p(\boldsymbol{z_\tau})] \end{aligned}$$

In terms of the maximum likelihood method and the non-negativity of KL divergence. We get the likelihood of trajectories as follows:

$$\begin{aligned} &\log p(\boldsymbol{\tau}) \geq \mathbb{E}_{\substack{\boldsymbol{z}_i\sim q_i(\boldsymbol{z}),\\ \boldsymbol{z_\tau}\sim q_{\boldsymbol{\tau}}(\boldsymbol{z})}}[\log p(\boldsymbol{\tau}|\boldsymbol{z}_i, \boldsymbol{z_\tau})] \\ &- KL[q_i(\boldsymbol{z})||p(\boldsymbol{z}_i)] - KL[q_{\boldsymbol{\tau}}(\boldsymbol{z})||p(\boldsymbol{z_\tau})] \end{aligned} \tag{3}$$

To maximize the likelihood above, we maximize the lower bound shown on the right of Equation 3. Therefore, we train the pre-trained model with the reconstructive objective on the right of Equation 3, in which the first term can be found as $\mathcal{L}_{recons}$ in our paper. The third term can be found as the $\mathcal{L}_{as}$ loss for agent-invariant network learning. Note that we replace the second term with discrete latent-variable models using a stop-gradient operator to update the corresponding parameters in $\mathcal{L}_{offline}$.

For the reconstruction objective, the decoder takes as input the concatenation of agent-specific and discrete hidden vectors to reconstruct the input that is to maximize the log-likelihood as follows:

$$\begin{aligned} &\mathcal{L}_{recons} = \mathbb{E}_{\substack{\boldsymbol{z}_i\sim q_i(\boldsymbol{z}),\\ \boldsymbol{z_\tau}\sim q_{\boldsymbol{\tau}}(\boldsymbol{z})}}[\log p(\boldsymbol{\tau}|\boldsymbol{z}_q(\boldsymbol{\tau}), \boldsymbol{z}_i)], \\ &\text{where } \boldsymbol{z}_q(\boldsymbol{\tau}) = \boldsymbol{e}_k,\ k = \arg\min_j \left\|\boldsymbol{z}_e(\boldsymbol{\tau}) - \boldsymbol{e}_j\right\|_2 \end{aligned} \tag{4}$$

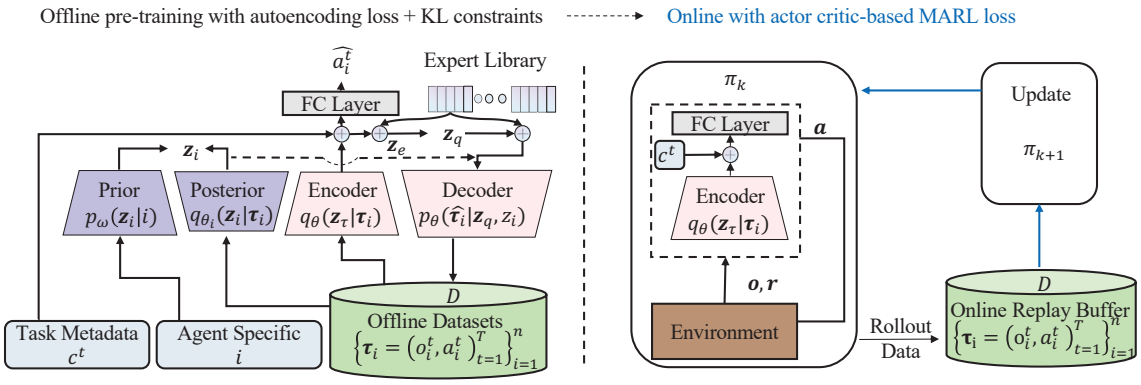


Figure 2: An overview of the M3 pipeline for multi-agent multi-task offline pre-training and online fine-tuning. The offline pre-trained model takes task metadata embeddings and offline trajectories as input to reconstruct the input and simultaneously predict the ground truth actions. Then the pre-trained policy is optimized with online interactions.

where $\boldsymbol{e}_j$ denotes the j_{th} item of the expert dictionary, z_e denotes the encoder output that equals to z_τ in Equation 3. Like the discrete latent variable method (VQ-VAE) [31], we optimize the expert dictionary with l_2 error moving the embedding vectors to the encoder outputs. Furthermore, the agent-invariant network is learned through an agent-specific objective $\mathcal{L}_{as} = KL[q_i(z)||p(z_i)]$ to minimize the divergence in hidden space. Additionally, we aim to map the encoded hidden into the ground truth actions a_i with supervised learning objective $\mathcal{L}_{SL} = \log p(a_i|z_e)$. Therefore, the pre-trained policy for each agent i can be represented as $\pi(\cdot|o_i, c) = p(a_i|z_e)$. Another term of objective in the offline phase is learning to quantize the embedding space into the codebook with stop-gradient techniques learned from the expert dictionary. The intuition of this design is to utilize an expert library across multi-task and multi-agent while the library is dynamic and learnable through large amounts of offline data pairs. However, the quantization operation is intractable due to the gradient clipping. We refer to the vector quantization method to update the embedding library with the stop gradient technique in Equation 5.

$$\mathcal{L}_{emb} = \|sg\left[z_e(\boldsymbol{\tau})\right] - \boldsymbol{e}\|_2^2 + \beta \left\|z_e(\boldsymbol{\tau}) - sg[\boldsymbol{e}]\right\|_2^2 \tag{5}$$

where sg is the stop-gradient operator defined as an identity at forwarding computation time and has zero partial derivatives, and β is a hyperparameter described in [31]. Therefore, our offline objective optimized with gradient ascent can be shown in Equation 6:

$$\mathcal{L}_{offline} = \mathcal{L}_{SL} + \mathcal{L}_{recons} + \mathcal{L}_{as} + \mathcal{L}_{emb} \tag{6}$$

The first term, $\mathcal{L}_{SL}$, is the supervised learning objective for policy pre-training, showing the performance on the offline dataset. $\mathcal{L}_{recons}$ is the reconstruction objective, which depicts how well the policy is represented in the latent space. $\mathcal{L}_{as}$ is the agent-specific objective that forces agents to behave differently for better coordination. The last term, $\mathcal{L}_{emb}$, is the embedding objective and is necessary to discretize policy representation, enabling agents to reuse parts of representations in each task. The four objectives aim to assign different policies to each agent under each task.

Online Fing-tuning. In this stage, we employ the pre-trained model to fine-tune it online as a multi-agent policy by sharing parameters. Yu et al. [52] conduct the multi-agent PPO by sharing actor and global critic to solve the multi-agent decision-making problem online. Therefore, we load the pre-trained model as the shared actor across agents to fine-tune it as the multi-agent PPO scheme. Multi-Agent PPO [52] leverage the CTDE framework to train a shared policy for each agent by maximizing the sum of their PPO-clip objectives as follows:

$$\sum_{i=1}^{n} \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\theta}_{\text{old}}},\mathbf{a}\sim\boldsymbol{\pi}_{\theta_{\text{old}}}} \left[\min\left(r, \text{clip}\left(r\right)_{1-\epsilon}^{1+\epsilon}\right)\hat{A}(\mathrm{s},\mathbf{a})\right] \tag{7}$$

where $r = \frac{\pi_{\boldsymbol{\theta}}(\mathrm{a}^i|\mathrm{s})}{\pi_{\boldsymbol{\theta}_{\text{old}}}(\mathrm{a}^i|\mathrm{s})}$ is the clip ratio, θ is initialized with the pre-trained policy parameter as the actor model, $\hat{A}$ denotes the advantage function computed by the actor and critic models. The clip operator aims at stabilizing the policy update.

4.2 Algorithm and Training Paradigm

This section aims to introduce each module in the pipeline and demonstrate its necessity.

Discrete Latent-Variable Models. To learn high-level transferable policy representations from trajectories, we leverage a discrete latent variable model. To learn a policy library that can vary with observations and specific tasks, we quantize continuous hidden vectors from the encoder based on task embeddings. Typically, vector quantization methods aim to learn discrete codebooks with stop-gradient operators, as the first term in Equation 5. The discrete representations can be seen as a modularization method for the multi-task setting like that in single-agent [50].

Task Prompting. To fine-tune our pre-trained model in the online stage following CMG, even on unseen tasks, we enforce the model to encode task meta information as the context both offline and online. Under a similar assumption in [57] that task embeddings have a stationary distribution across tasks, the multi-task model can be optimized with context representation learning. Moreover, learning context representation is an exciting topic in single-agent offline methods like Prompt DT [49]. We will explore it in the future in multi-agent cases. In the final version of M3, we use the semantic task description as the context to improve generalization.

Agent Invariant Module. By introducing the agent invariant module, we hypothesize that policy representation across multiple agents can be split into agent sharing and specific. As the example of UAV and football games in the introduction, we suppose that

agent-specific is necessary for heterogeneous environments such as SMAC and MPE. The pre-trained policy should be learned by splitting the latent representations. Empirically, we utilize the maximum likelihood method to derive the modified reconstruction loss. Therefore, we give this loss after introducing the agent-specific properties with ELBO bounds by minimizing the marginal probability and Kullback-Leibler divergence as shown in Equation 3, where we need to specify the agent-specific hidden vector $\boldsymbol{z}_i$ of agent i. We show more detailed derivative in Appendix B.

AWAC. To fix the data distribution shift between offline and online, Nair et al. [27] derives an objective with the Karush Kuhn Tucker(KKT) conditions with $D_{KL}(\pi_\theta \| \pi_\beta) \leq \epsilon$, where π_θ is the actor being updated and π_β represents the behavior policy generated the previous offline data. We utilize this trick and show the necessity with a guarantee bound in Section 5. Results of the ablation study in Section 6 also validate its benefits to fix the bootstrap error accumulation as observed in some prior works [23, 25] for fine-tuning the pre-trained policy online well.

5 THEORETICAL ANALYSIS

We use empirical risk minimization (ERM) in the pre-training stage to fit the experience distribution in the offline dataset $\mathcal{D} = \cup_c \mathcal{D}_c$ with a sharing policy across agents $\hat{\pi}(a|o, c)$, where each task-specific dataset $\mathcal{D}_c$ is collected from a behavior policy π_β. In the offline pre-training stage, we denote the return $R(s, \pi_\beta(\cdot|s))$ as $R_\beta(\boldsymbol{\tau})$. Furthermore, the multi-task policy $\hat{\pi}(a|o, c)$ is retrained online on another task. In offline pre-training MT-MARL setting, the pre-trained policy is fine-tuned on a seen or unseen task to maximize the corresponding discounted return $R(\boldsymbol{\tau}, c^{test}) = \sum_t \gamma^t r^{c^{test}}(s^t, \boldsymbol{a}^t)$, where $\boldsymbol{a} := \{a_i \sim \hat{\pi}'(a_i|o_i, c^{test})\}_{i=1}^n$, $o_i = O(s, i)$. Moreover, we derive the generalization gap as: $\Phi = \left| \frac{1}{N} \sum_{j=1}^N R(\boldsymbol{\tau}^j, c^{train}) - \mathbb{E}_{\boldsymbol{\tau} \sim \mathcal{D}'_{\hat{\pi}}} R(\boldsymbol{\tau}, c^{test}) \right|$. The gap can be upper-bounded by

$$
\begin{aligned}
2\Phi \leq & \underbrace{\left| \frac{1}{N} \sum_{j=1}^N R(\boldsymbol{\tau}^j, c^{train}) - \mathbb{E}_{\boldsymbol{\tau} \sim \mathcal{D}} R_\beta(\boldsymbol{\tau}) \right|}_{\text{offline gap}} + \\
& \underbrace{\left| \frac{1}{N} \sum_{j=1}^N R(\boldsymbol{\tau}^j, c^{train}) - \mathbb{E}_{\boldsymbol{\tau} \sim \mathcal{D}_{\hat{\pi}}} R(\boldsymbol{\tau}, c^{test}) \right|}_{\text{intrinsic error}} \\
& + \underbrace{\left| \mathbb{E}_{\boldsymbol{\tau} \sim \mathcal{D}} R_\beta(\boldsymbol{\tau}) - \mathbb{E}_{\boldsymbol{\tau} \sim \mathcal{D}'_{\hat{\pi}}} R(\boldsymbol{\tau}, c^{test}) \right|}_{\text{online gap}} \\
& + \underbrace{\left| \mathbb{E}_{\boldsymbol{\tau} \sim \mathcal{D}_{\hat{\pi}}} R(\boldsymbol{\tau}, c^{test}) - \mathbb{E}_{\boldsymbol{\tau} \sim \mathcal{D}'_{\hat{\pi}}} R(\boldsymbol{\tau}, c^{test}) \right|}_{\text{external error}}
\end{aligned}
\tag{8}
$$

using the triangle inequality, where c^{train} denotes the task contexts in the training data, c^{test} denotes the online task on which the policy is to be fine-tuned. The offline gap emerges when the offline pre-trained policy π_θ converges under c^{train} according to Equation 6. We deal with the online gap term via AWAC. The other intrinsic and external error terms result from online policy $\hat{\pi}_\theta$ based on Equation 7 facing the intrinsic randomness of the environment and the transition distribution shift in terms of different contexts. Especially, when it comes to online MTRL without offline pre-training, the generalization gap above becomes intrinsic and external errors shown in recent work [46]. We give more detailed theoretical derivatives in Appendix B.

6 EXPERIMENTAL EVALUATION

Our experiments aim at demonstrating the effectiveness of M3 on offline pre-training MT-MARL in few-shot and even zero-shot settings. The offline multi-task datasets are collected from the running policy, MAPPO [52], on the well-known SMAC task [36], released by [26]. This offline dataset contains millions of timesteps from all subtasks in the challenging SMAC, which is appropriate for the multi-task setting. Each dataset contains amounts of trajectories: $\boldsymbol{\tau} := (s^t, o^t, \boldsymbol{a}^t, r^t)_{t=1}^T$. All experiments are developed with ten different random seeds.

Baselines. In offline multi-task multi-agent reinforcement learning settings, few well-matched comparative methods for baseline selection are available. To this end, we modify SOTA in online single-agent multi-task (CARE [39]) and offline multi-agent single-task (MADT [26]) to fit this setting.

- MAPPO [52]: An multi-agent extension of PPO [37] algorithm by setting the sharing actor taking the local observation of each agent, and sending the global state into a centralized critic to optimize the policy as PPO-style. We compare our method with it as an online MARL baseline.
- CARE [39]: **C**ontextual **A**ttention-based **RE**presentation learning (CARE) proposes to share contextual encoding across tasks in the single-agent case. We extend it by sharing policies across agents for our multi-agent setting as an online MT-MARL baseline.
- MADT [26]: Multi-Agent Decision Transformer (MADT) has shown high performance in [26, 43] with extensive results in offline pre-training including Fill-In, Equal Space, GridWorld, Highway, and SMAC. We pad the state and action space across tasks with zero for the offline MT-MARL setting.

In addition, in Appendix E, we set another method, UPDeT [19]: Universal Policy Decoupling Transformer (UPDeT), which utilizes transformer-based value networks to tackle the adaptations among various inputs, as online MT-MARL baseline. Our method outperforms its sample efficiency online on several SMAC maps.

Implementation details. We train a MAPPO [52] policy for one task and compare it with our method fine-tuned on that task. We develop experiments on the challenging SMAC with multiple tasks, which is suitable for offline pre-training MT-MARL. In addition, the information regarding computational resources used is Enterprise Linux Server with 256 CPU cores and 1 NVIDIA A100 GPU (40G memory). More implementation details, such as offline episode number, training epoch, and hyperparameters used in our experiments, can be found in Appendix D.

Table 1: *JP*/*TT*: Jump point (*JP*: the initial performance/returns of agents online) and Time to Threshold (*TT*: the extra rollout timesteps needed to reach the certain return threshold 18.0 online)) results of online and offline MARL/MTRL algorithms on five easy and mixed tasks from SMAC. The ◊ and △ is black denotes the method whether is multi-task and offline or not.

Method	Type	2m vs. 1z (easy)	2s vs. 1sc (easy)	3m (easy)	3s vs. 3z (easy)	3s vs. 4z (easy)
MAPPO [52]	◊△	/1.5 (±0.4)e5	/8.3 (±0.6)e4	/4.0 (±1.5)e5	/3.2 (±0.2)e5	/1 (±0.4)e6
CARE [39]	♦△	5.2 (±0.8)/∞	15.8 (±2.5)/1 (±0.5)e6	7.4 (±1.9)/∞	8.2 (±2.2)/∞	5.4 (±3.3)/∞
MADT [26]	♦▲	17.6 (±0.8)/1 (±0.3)e3	18.3 (±0.2)/0	18.8 (±0.5)/0	17.7 (±0.5)/1.9 (±1.1)e4	18.4 (±0.1)/0
M3 (ours)	♦▲	**20.0/0**	**20.0/0**	**19.6 (±0.2)/0**	**17.7 (±0.3)/1.9 (±0.5)e3**	**20.0/0**
Method	Type	3m (easy)	8m (easy)	3s vs. 5z (hard)	MMM (easy)	MMM2 (super hard)
MAPPO [52]	◊△	/4 (±0.5)e5	/6.4 (±0.8)e5	/1.1 (±0.2)e6	/2.1 (±0.3)e6	/1.2 (±0.5)e7
CARE [39]	♦△	17.7 (±1.8)/6.4 (±0.4)e5	10.9 (±0.5)/∞	4.1 (±0.3)/∞	1.9 (±0.3)/∞	1.4 (±0.5)/∞
MADT [26]	♦▲	9.5 (±2.6)/7.9 (±0.4)e4	6.7 (±0.6)/1.8 (±2.2)e5	5.4 (±1.3)/2.6 (±0.5)e6	9.1 (±2.4)/1.2 (±1.5)e5	7.5 (±1.7)/∞
M3 (ours)	♦▲	**20.0/0**	**20.0/0**	**11.8 (±1.3)/6.8 (±1.8)e4**	**16.2 (±0.8)/5.5 (±1.2)e4**	**11.3 (±1.2)/1.5 (±0.1)e6**

6.1 How M3 compares with existing online and offline MTRL baselines in few-shot settings?

To validate the sample efficiency of the proposed M3 method in the multi-task setting, we compare it with some existing baselines. Recent works have studied MTRL mainly in two ways: **i)** online learning with the modularized structure, knowledge transfer, or shared backbone with a task-specific head; **ii)** offline pre-training from collected multi-task data and fine-tuning the pre-trained policy on the downstream task online. In this section, we compare M3 with state-of-the-art methods specifically extending in multi-agent setup from offline to online. From the online perspective, we compare a well-known state-of-the-art MARL algorithm called MAPPO [52] to validate the effectiveness of the pre-trained model. In addition, [39] shows the supreme performance on the online single-agent MTRL benchmark meta-world [55]. Here we extend the single-agent MTRL algorithm by sharing the policy structure across multiple agents, and the task metadata is replaced with the task information in our SMAC tasks. Besides, we compare our method with MADT [26] by similarly taking multi-task offline data to show the benefit of M3 in the offline MT-MARL setting. We validate M3 effectiveness compared with the baselines mentioned above based on two sets of experiments. In the first set, we firstly pre-train M3 and MADT on offline trajectories from five easy maps (`2m_vs_1z, 2s_vs_1sc, 3m, 3s_vs_3z, 3s_vs_4z`) provided by MADT. We then continuously fine-tune the pre-trained policies on each of these five subtasks. The second set includes five subsets from maps with easy, hard, and super-hard difficulty (`3m, 8m, 3s_vs_5z, MMM, MMM2`), and fine-tunes the pre-trained policy similarly as above. The evaluation metrics mainly focus on the $\mathcal{JP}$ and *TT* described in Table 1. Note that we do not record $\mathcal{JP}$ of MAPPO because the model is trained from scratch on each task as a single-task method. Table 1 shows our method can outperform baselines not only on the $\mathcal{JP}$ but the convergence speed online on *TT* on each subtask. As described in the table caption, we set Time-to-threshold to measure the online sample efficiency. We choose 18.0 for two reasons: 1) The threshold is determined based on the SMAC domain. The reward is normalized in 0∼20 and proportional to the win rate. 2) The expert data is divided with the reward from 18.0 to 20. Since 18.0 is very close to 20, we suppose M3 could still outperform the asymptotic performance by setting the threshold as 20. That indicates our method can improve the online MARL sample efficiency in the multi-task setting with explicit policy representation learning.

6.2 Can the pre-trained policy be generalized to unseen tasks?

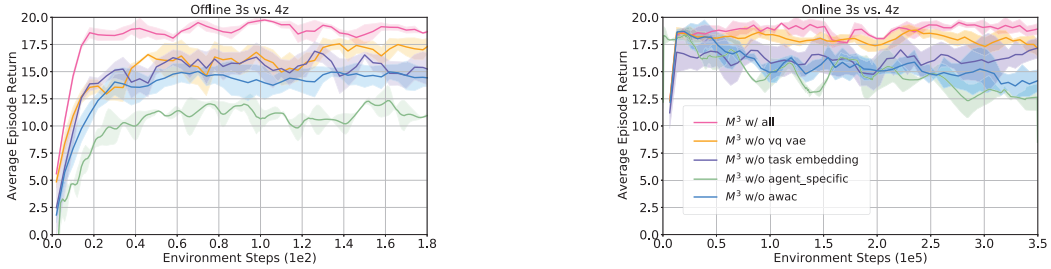
Table 2: *JP/TT* described in Table 1 when the algorithm is fine-tuned on the following unseen maps (3m, and MMM2).

Method	3s_vs_4z (easy)	MMM2 (super hard)
CARE [39]	8.3 (±1.3)/∞	1.4 (±0.4)/∞
MADT [26]	4.2 (±1.2)/7.0 (±0.6)e5	8.6 (±2.3)/∞
M3 (ours)	**15.3 (±1.1)/4.6 (±0.9)e5**	**12.5 (±2.6)/1.3 (±0.2)e6**

To validate the generalization ability of M3 on unseen tasks in the multi-task setting, we fine-tune the pre-trained policy on unseen maps in which the policy is trained on four previously determined maps. MAPPO in the baselines above is not considered due to the single-task reason. In practice, eight maps are selected from two difficulty levels (easy and mixture). In terms of similarity, the test map can be close or not to the training map. In this subsection, we pre-train two policies (M3-easy, M3-mixture) on eight maps (4 maps per difficulty), and fine-tune them on another held-out map. To test these pre-trained policies, we set `3s_vs_4z` and `MMM2` aside for fine-tuning M3-easy and M3-mixture respectively. Table 2 shows M3 can generalize to unseen tasks better than other online and offline multi-task baselines.

6.3 What the policy library learns?

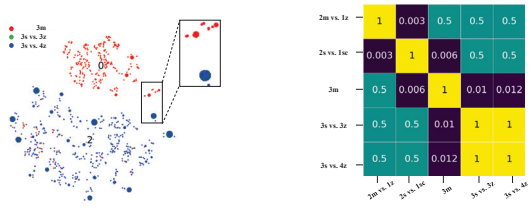
To determine whether the policy representation in M3 can discriminate different tasks and execute distinct policies automatically, we show the two sets mentioned in Section 6.1 of task similarity based on task embedding of the map information in Figure 4. Furthermore, we visualize our encoded hidden vectors when inferred from different maps. We leverage the pre-trained M3-easy policy for rollout with different map information (`3m`, `3s_vs_3z`, and `3s_vs_4z` for



(a) (b)

Figure 3: Ablation experiments to validate the necessity of each module in M3 with offline evaluation and online fine-tune returns. In this experiment, the pre-trained policy is trained from the five easy maps sed in Section 6.

M3-easy). We note that the scatters shown in Figure 4 indicate that M3 can discriminate different tasks by task information prompt. The visualization in Figure 4(a) shows our model can discriminate the different tasks in various latent spaces. The similarity factor causes the super close representations between them in Figure 4(b). In addition, similar tasks, such as `3s_vs_3z` and `3s_vs_4z`, have learned almost the same policy representation compared with other tasks (e.g., `3m`). There is also a little overlap between red and blue points. We hypothesize that different tasks may contain similar optimal strategies in the interactions, and our model can discriminate different tasks in absolutely most cases. We want to explore this overlap as a degree of multi-task policy representation in the future.



(a) (b)

Figure 4: (a) Policy representations of M3 on three tasks. (b) Task similarity measured with the cosine distance of five easy task embeddings.

6.4 Ablation Study

In this section, we validate the necessity of each module in our method. To answer the research questions: **RQ1:** Can VQ-VAE help multi-task learning in multi-agent offline pre-training? **RQ2:** Can the task embedding improve the performance in multi-task settings? **RQ3:** Is the agent-invariant network necessary? **RQ4:** Does awac work for online fine-tuning? We consider four types of ablations: (1) Remove the reconstruction loss based on vector quantized VAE and replace the pre-train policy structure with causal transformers, where the removed policy structure is the same as the MADT that shares a policy across agents; (2) Remove the injection of task embeddings in the encoder where the pre-trained policy does not have offline or online access to any task-specific information; (3) Remove the agent-invariant part of the hidden space, where the hidden space is used to predict agent actions rather than split into two parts (agent-sharing and specific); (4) Remove the awac to test the online fine-tuning performance, which we hypothesize it benefits the online policy close to the offline pre-trained policy. In Figure 3(a) and 3(b), we show M3 variants comparison results with above modifications. The discrete representations and context with task embedding are effective, especially in the online fine-tuning stage. Moreover, the awac and agent-specific modules are important for offline and online performance. In particular, after removing awac and agent-specific modules, the fine-tuning average return online decreases over time. We suppose this phenomenon is caused by the bootstrap error from the offline and online data mismatch, which harms the online fine-tuning process.

7 CONCLUSION

In this work, we propose to solve the offline multi-task multi-agent reinforcement learning problem by modularizing offline multi-agent pre-training M3, a representation learning approach for pre-training policy across diverse tasks. M3 leverages context-based methods to reconstruct inputs and predict actions while learning policy representations in offline data. In order to fit the multi-task setting, we use vector quantization techniques and discrete policy representations in the hidden space. To tackle the non-stationarity and heterogeneity of multiple agents, we induce the agent-specific module to improve intra-agent transfer. Furthermore, we use task metadata to prompt generalization to unseen tasks. We theoretically analyze the generalization error for fine-tuning our pre-trained policy. Empirically, regardless of the online and offline settings of the StarCraftII Multi-Agent Challenge, the multi-task pre-trained policy outperforms the state-of-the-art algorithm in terms of effectiveness and generalization on various maps with different difficulties. Moreover, our pre-trained policy shows that it can discriminate tasks from hidden policy representations. We believe this is the first solution specifically designed for the offline MT-MARL problem and hope to inspire more work in this practical setting. Future research may consider improving the adaptation error analyzed in Equation 8 when the pre-trained policies are learned from suboptimal offline data. We will also investigate a suitable measurement as the degree of multi-task policy learning.

ACKNOWLEDGMENTS

This work is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (No. XDA27030300) and the Program for National Nature Science Foundation of China (62073324).

REFERENCES

[1] Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. 2021. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *arXiv preprint arXiv:2101.05265* (2021).

[2] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew Taylor. 2014. Online multi-task learning for policy gradient methods. In *International conference on machine learning*. PMLR, 1206–1214.

[3] Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*. PMLR, 166–175.

[4] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2006.11477* (2020).

[5] Diana Borsa, Thore Graepel, and John Shawe-Taylor. 2016. Learning shared representations in multi-task reinforcement learning. *arXiv preprint arXiv:1603.02041* (2016).

[6] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345* (2021).

[7] Li Chenghao, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. 2021. Celebrating diversity in shared multi-agent reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021), 3991–4002.

[8] Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796* (2020).

[9] Jingjing Cui, Yuanwei Liu, and Arumugam Nallanathan. 2019. Multi-agent reinforcement learning-based resource allocation for UAV networks. *IEEE Transactions on Wireless Communications* 19, 2 (2019), 729–743.

[10] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. 2019. Sharing knowledge in multi-task deep reinforcement learning. In *International Conference on Learning Representations*.

[11] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. 2017. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2169–2176.

[12] Wei Du and Shifei Ding. 2020. A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications. *Artificial Intelligence Review* (2020).

[13] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. 2020. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219* (2020).

[14] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*. PMLR, 2052–2062.

[15] Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. 2018. Learning policy representations in multiagent systems. In *International conference on machine learning*. PMLR, 1802–1811.

[16] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. 2019. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603* (2019).

[17] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. 2020. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193* (2020).

[18] Assaf Hallak, Dotan Di Castro, and Shie Mannor. 2015. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259* (2015).

[19] Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. 2021. Updet: Universal multi-agent reinforcement learning via policy decoupling with transformers. *arXiv preprint arXiv:2101.08001* (2021).

[20] Rahul Jain, Preeti Ranjan Panda, and Sreenivas Subramoney. 2017. Cooperative multi-agent reinforcement learning-based co-optimization of cores, caches, and on-chip network. *ACM Transactions on Architecture and Code Optimization (TACO)* 14, 4 (2017), 1–25.

[21] Michael Janner, Qiyang Li, and Sergey Levine. 2021. Reinforcement Learning as One Big Sequence Modeling Problem. *arXiv preprint arXiv:2106.02039* (2021).

[22] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. *Advances in Neural Information Processing Systems* 32 (2019), 11784–11794.

[23] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems* 32 (2019).

[24] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[25] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[26] Linghui Meng, Muning Wen, Yaodong Yang, Chenyang Le, Xiyun Li, Weinan Zhang, Ying Wen, Haifeng Zhang, Jun Wang, and Bo Xu. 2021. Offline Pre-trained Multi-Agent Decision Transformer: One Big Sequence Model Conquers All StarCraftII Tasks. *arXiv preprint arXiv:2112.02845* (2021).

[27] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. 2020. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359* (2020).

[28] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. 2017. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2661–2670.

[29] Frans A Oliehoek and Christopher Amato. 2016. *A concise introduction to decentralized POMDPs*. Springer.

[30] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. 2017. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International Conference on Machine Learning*. PMLR, 2681–2690.

[31] Aaron Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2019. Neural Discrete Representation Learning. (2019).

[32] Sherjil Ozair, Yazhe Li, Ali Razavi, Ioannis Antonoglou, Aaron van den Oord, and Oriol Vinyals. 2021. Vector Quantized Models for Planning. *arXiv: Learning* (2021).

[33] Adolfo Perrusquía, Wen Yu, and Xiaoou Li. 2021. Multi-agent reinforcement learning for redundant robot control in task-space. *International Journal of Machine Learning and Cybernetics* 12, 1 (2021), 231–241.

[34] Roxana Rădulescu, Patrick Mannion, Diederik M Roijers, and Ann Nowé. 2020. Multi-objective multi-agent decision making: a utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems* 34, 1 (2020), 1–52.

[35] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with vq-vae-2. In *Advances in neural information processing systems*. 14866–14876.

[36] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. 2019. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043* (2019).

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[38] Shagun Sodhani, Franziska Meier, Joelle Pineau, and Amy Zhang. 2021. Block Contextual MDPs for Continual Learning. *arXiv preprint arXiv:2110.06972* (2021).

[39] Shagun Sodhani, Amy Zhang, and Joelle Pineau. 2021. Multi-Task Reinforcement Learning with Context-based Representations. *arXiv preprint arXiv:2102.06177* (2021).

[40] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. 2020. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136* (2020).

[41] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. 2021. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*. PMLR, 9870–9879.

[42] Fumihide Tanaka and Masayuki Yamamura. 2003. Multitask reinforcement learning on the distribution of MDPs. In *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation.*, Vol. 3. IEEE, 1108–1113.

[43] Wei-Cheng Tseng, Tsun-Hsuan Wang, Yen-Chen Lin, and Phillip Isola. 2022. Offline Multi-Agent Reinforcement Learning with Knowledge Distillation. In *Thirty-Sixth Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=yipUuqxveCy

[44] Wei-Cheng Tseng, Wei Wei, Da-Chen Juan, and Min Sun. 2021. Meta-cpr: Generalize to unseen large number of agents with communication pattern recognition module. *arXiv preprint arXiv:2112.07222* (2021).

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[46] Huan Wang, Stephan Zheng, Caiming Xiong, and Richard Socher. 2019. On the generalization gap in reparameterizable reinforcement learning. In *International Conference on Machine Learning*. PMLR, 6648–6658.

[47] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. 2020. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. In *International Conference on Machine Learning*. PMLR, 9876–9886.

[48] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. 2020. RODE: Learning Roles to Decompose Multi-Agent Tasks. In *International Conference on Learning Representations*.

[49] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. 2022. Prompting decision transformer for few-shot policy

generalization. In *International Conference on Machine Learning*. PMLR, 24631–24645.

[50] Ruihan Yang, Huazhe Xu, YI WU, and Xiaolong Wang. 2020. Multi-Task Reinforcement Learning with Soft Modularization. *Advances in Neural Information Processing Systems* 33 (2020), 4767–4777.

[51] Yiqin Yang, Xiaoteng Ma, Li Chenghao, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. 2021. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021).

[52] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. 2021. The surprising effectiveness of mappo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955* (2021).

[53] Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn. 2021. Conservative data sharing for multi-task offline reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021).

[54] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient Surgery for Multi-Task Learning. *Advances in Neural Information Processing Systems* 33 (2020).

[55] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. 2020. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*. PMLR, 1094–1100.

[56] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. 2020. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742* (2020).

[57] Tony Z Zhao, Jianlan Luo, Oleg Sushkov, Rugile Pevceviciute, Nicolas Heess, Jon Scholz, Stefan Schaal, and Sergey Levine. 2021. Offline meta-reinforcement learning for industrial insertion. *arXiv preprint arXiv:2110.04276* (2021).

[58] Yi Zhao, Haoyu Li, Cheng-I Lai, Jennifer Williams, Erica Cooper, and Junichi Yamagishi. 2020. Improved prosody from learned f0 codebook representations for vq-vae speech waveform reconstruction. *arXiv preprint arXiv:2005.07884* (2020).

[59] Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, et al. 2020. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. *arXiv preprint arXiv:2010.09776* (2020).