Sharing Experience in Multitask Reinforcement Learning

Tung-Long Vuong *, **Do-Van Nguyen**, **Tai-Long Nguyen**, **Cong-Minh Bui**, **Hai-Dang Kieu**, **Viet-Cuong Ta**, **Quoc-Long Tran** and **Thanh-Ha Le**
HMI Lab, UET, Vietnam National University, Hanoi, Vietnam
{longvt94, ngdovan, tailongnguyen97, buicongminh2717, dangkh109, tqlong, lthavnu}@gmail.com, cuongtv@vnu.edu.vn

Abstract

In multitask reinforcement learning, tasks often have sub-tasks that share the same solution, even though the overall tasks are different. If the shared-portions could be effectively identified, then the learning process could be improved since all the samples between tasks in the shared space could be used. In this paper, we propose a Sharing Experience Framework (SEF) for simultaneously training of multiple tasks. In SEF, a confidence sharing agent uses task-specific rewards from the environment to identify similar parts that should be shared across tasks and defines those parts as shared-regions between tasks. The shared-regions are expected to guide task-policies sharing their experience during the learning process. The experiments highlight that our framework improves the performance and the stability of learning task-policies, and is possible to help task-policies avoid local optimums.

1 Introduction

Reinforcement learning (RL) [Sutton and S, 1988; Watkins and Dayan, 1992] has long been an interesting topic in robotics and artificial intelligent (AI) domains. RL enables AI agents to learn appropriate policies through interacting with a given environment. In traditional RL approaches [Sutton and Barto, 1998], such as Q-learning and SARSA, linear approximate functions are used to solve the generalization of machine learning algorithms in infinite and dynamic scenarios [Grounds and Kudenko, 2008; Konidaris *et al.*, 2011].

Recently, deep learning (DL) approaches have obtained significant success in various research areas, including vision and natural language processing applications [LeCun *et al.*, 2015]. The DL approaches use a huge number of auto-learned parameters and a variety of network architectures, such as convolutional neural networks (CNN) [Krizhevsky *et al.*, 2012], and long-short term memory (LSTM) [Graves *et al.*, 2013] networks. These deep neural networks (DNN) are able to extract unprecedented features and increase generalization capability. In reinforcement learning framework, Deep reinforcement learning (DRL), a combination of RL and DL, allows RL agents to converge in training with complex tasks that require both feature and state representation and improve overall performance significantly. Well-known research using DRL includes [Mnih *et al.*, 2015] and [Mnih *et al.*, 2016] in playing Atari and Go games respectively.

With different scenarios in the problem of decision making, agents will definitely involve in different action spaces [Lillicrap *et al.*, 2015; Mnih *et al.*, 2016], and face with various physical factors or surrounding views. When the environment becomes more complex and infinite, the problem of decision making in DRL approaches will also become more challenging. An open research question raised is that how to reuse common behaviors in similar scenarios? If we are able to transfer knowledge between similar environments, it will reduce the training time for agents, and then the difficulty in applying RL methods for the dynamic and infinite environment can simply be solved.

One way to deal with multi-task problems is to use unique policy and train with different tasks simultaneously [Yang *et al.*, 2017; Teh *et al.*, 2017] or sequentially [Rusu *et al.*, 2016] to update both policy or/and value approximation functions. However, those approaches do not allow capturing the difference and the similarity among the tasks.

In more complex problems, hierarchical reinforcement learning methods [Andreas *et al.*, 2016; Devin *et al.*, 2017] may solve simple sub-tasks then combine those sub-tasks to solve the complex tasks. Transfer learning [Rusu *et al.*, 2015; Parisotto *et al.*, 2015] is also used to train agent policies in some tasks and then adjust to new ones in multi-task scenarios. Learning skills in some new studies [Borsa *et al.*, 2016; Gupta *et al.*, 2017; Hausman *et al.*, 2018; Zhang *et al.*, 2017] may be a means to capture embedding feature spaces, then those features are utilized to solve different tasks. However, those approaches always have to consider how to define sub-task that relates to the main tasks.

In this paper, we propose a new framework for the simultaneous RL problem of multiple tasks. Our proposed framework enables agents to explore similar parts between tasks in an environment to transfer experience across all tasks during the training process. To our knowledge, there is no method that has discovered shared parts to enhance policy update with multiple tasks instead of updating the main policy and transferring to sub-tasks. Our main contribution can be summa-

*longvt94@gmail.com

rized as follows:

1. Proposing a sharing confidence agent that uses data samples from task-policies to identifying shared portions among tasks.

2. Proposing a sharing experience framework allow task-policies sharing samples in shared portions to make the learning process more stable and robust.

3. We empirically evaluate the framework in different scenarios to show that it is possible to discover shared portions among tasks as latent representations and use those latent representations to reduce training time of the tasks.

2 Related Work

Usually, to learn different tasks, an agent can be trained directly with multiple tasks simultaneously to update policy or/and value approximation functions. The authors in [Yang *et al.*, 2017] propose a multi-actor and single-critic architecture to implicitly exploit the shared visual features to tackle multi-task learning. Research in [Teh *et al.*, 2017] uses distilled policy centroid of all task policies to obtain common knowledge and transfer it among the training of policies for different tasks by constraining the distance between distilled policy and task-policies.

Another approach is continual learning where the agent is trained on multiple tasks sequentially. A special network architecture called Progressive Neural Network is proposed in [Rusu *et al.*, 2016], which explicitly distills the learned experiences and representations of previous well-trained models to a new one while also accounting the problem of forgetting. However, this idea suffers from the quadratic growth of the number of model parameters and the difficulty in choosing the optimal inference order.

To solve complex tasks, agents can be trained to achieve different sub-goals before completing a final task in some researches. Hierarchical RL uses modularity to achieve transfer learning for robotic tasks. [Devin *et al.*, 2017] decomposes policy neural networks into task-specific or agent-specific modules and then combine different components across agents and tasks to pave the way for new training. The authors in [Andreas *et al.*, 2016] also decompose task-specific policies into sub-policies, in which some are shared across tasks using prior sketch annotations and jointly maximize the expected discounted rewards overall policies. Curriculum learning is also utilized to further facilitate training in complex environments. A hierarchical Bayesian model [Wilson *et al.*, 2007] is used to estimate the distribution over different environments and correspondingly induced MDPs, which may be reused to accelerate the training process by the agent jumping into environments that it has never encountered before.

Transfer learning and distillation are also of broad interest in multi-task RL. Distillation approach combines multiple expert policies into one by applying supervised regression over the action distribution of distilled policy and expert policies [Rusu *et al.*, 2015; Parisotto *et al.*, 2015]. However, these algorithms require expert policies not to present conflicting actions.

Another strategy is to learn shared skills or features across tasks, which can later transfer to new tasks. Studies in [Borsa *et al.*, 2016; Zhang *et al.*, 2017] introduce the idea of integrating successor features induced from the neural encoder into multi-task learning framework, in which a linear dependency between representations across shared structure environments is assumed to enable knowledge to transfer from trained policy networks. Invariant feature representations can also be learned [Gupta *et al.*, 2017] to transfer information about a skill that could be shared even across different robot architectures. However, the approach relies on prior information in proxy tasks to determine the invariant spaces. To explore important skills, [Hausman *et al.*, 2018] proposed reinforcement learning method that uses an embedding space defined by latent variables, entropy-regularized policy gradient and variational inference to learn a minimum number of distinct skills that are necessary to solve a given set of tasks. These learned skills can be interpolated and/or sequenced in order to solve more complex tasks. Naturally, these aforementioned approaches are similar to transfer learning methods.

3 Sharing Experience Framework

In our proposed SEF, a confident-sharing-agent Z_ϕ, which starts without a preliminary knowledge of shared regions across tasks, is co-trained with task-polices π_{θ_i} in an iterative process such that the sharing-agent provides shared regions to the task-policies. Task-policies' samples are shared in their shared regions to improve the learning process. The new data samples are generated from the updated task-policies. Then, the sharing-agent observing task-specific rewards from the environment which analyzed from task-policies' new samples to refine the prediction of shared regions. In this way, sharing-agent Z_ϕ and task-policies π_{θ_i} mutually reinforce by providing the best response to the other.

3.1 Preliminaries

We define a discrete-time finite-horizon discounted Markov decision process (MDP) by a tuple $M = \{S, A, P, r, \rho_0, \gamma, T\}$, in which S is a state set, A is an action set, $P : S \times A \times S \to \mathbb{R}^+$ is a transition probability distribution, $r : S \times A \to [-R_{max}, R_{max}]$ is a bounded reward function, $\rho_0 : S \to \mathbb{R}_+$ is an initial state distribution, $\gamma \in [0, 1]$ is a discount factor, and T is the horizon. In policy search methods, the stochastic policy $\pi_\theta : S \times A \to \mathbb{R}_+$ is typically optimized by maximizing its expected discounted return with respect to θ:

$$\eta(\pi_\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \quad (1)$$

where $\tau = (s_0, a_0, ...)$ denotes the whole trajectory, $s_0 \sim \rho_0(s_0)$, $a_t \sim \pi_\theta(a_t \mid s_t)$, and $s_t \sim P(s_{t+1} \mid s_t, a_t)$. The policy gradient methods maximize the expected total reward by repeatedly rolling out trajectories $\tau = (s_0, a_0, ...)$ from policy π_θ that is equal to maximize the objective function:

$$J(\theta) = \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)} \left[\pi_\theta(a_t|s_t) \Psi(s_t, a_t) \right] \quad (2)$$

Where Ψ_t could be one of the followings: total reward of trajectory $\sum_{t=0}^{\infty} r_t$, total reward following action $\sum_{t'=t}^{\infty} r_{t'}$,

state-action function $Q^{\pi}(s_t, a_t)$, and advantage function $A^{\pi}(s_t, a_t)$.

For this work, we consider a multi-task RL setting, where there are N tasks; and that the action space A, the state space S, and the transition dynamics $p_i(s' \mid s, a)$ are the same across all tasks $i = 1, 2, \ldots, N$. Let $r_i(s, a)$ and π_{θ_i} be the task-specific reward function and the stochastic policy for task i respectively.

3.2 Sharing Experience across Task-policies

This section describes a method that allows task-policies share the experience when they know their shared regions. We assume a existence of **a probabilistic sharing agent** Z_ϕ that takes (s, i, j) as input where s is state, i and j are task indexes and outputs a probability of state s belonging to shared regions of task i and j. A sharing action z_s is sampled from $Z_\phi(\cdot|s, i, j)$ and receives value of $\{1 : \text{share}, 0 : \text{non-share}\}$ presenting the decision of agent Z_ϕ.

The sharing-experience objective function is constructed to allow task-policies to share the experience in their shared regions as follows:

$$J_{\text{SEF}}(\theta_i) \approx \mathbb{E}\left[\min\left\{\begin{matrix}\frac{\pi_{\theta_i}}{W_i} * \Psi_k, \\ \text{clip}(\frac{\pi_{\theta_i}}{W_i}, 1-\epsilon, 1+\epsilon) * \Psi_k)\end{matrix}\right\}\right] \quad (3)$$

with full notation of: $\mathbb{E}[...]$ is $\mathbb{E}^{k:=1\mapsto N}_{(s_{k,t},a_{k,t})\sim P_{\theta_k}(s_{k,t},a_{k,t})}[...]$, π_{θ_i} is $\pi_{\theta_i}(a_{k,t}|s_{k,t})$, W_i is $W_i(a_{k,t}|s_{k,t})$, Ψ_k is $\Psi_k(a_{k,t}|s_{k,t})$ and

$$W_i(a|s) = \frac{\sum_k^n z_{s_{k,t}} \pi^{\text{old}}_{\theta_k}(a|s)}{\sum_k^N z_{s_{k,t}}} \quad (4)$$

where:

- The notation $\mathbb{E}^{k:=1\mapsto N}_{(s_{k,t},a_{k,t})\sim P_{\theta_k}(s_{k,t},a_{k,t})}[...]$ indicates the expected value of samples $(s_{k,t}, a_{k,t})$ generated from distributions $\{P_{\theta_k}|k := 1 \mapsto N\}$ and satisfied the constraint that state $s_{k,t}$ must belong to shared regions of task i and k: $\left[z_{s_{k,t}} \sim Z_\phi(\cdot|s_{k,t}, i, k)\right] = 1$.
- $z_{s_{k,t}}$ is sharing action which is sampled from probabilistic function $Z_\phi(\cdot|s_{k,t}, i, k)$. If a state s belongs to shared regions, samples of task-policy π_{θ_j} at the state s will be aggregated to update policy π_{θ_i} via important sampling weight. We believe that combining samples of task-policies in their shared regions in this way can enhance data-efficiency over task-policies.

  In general, task-policies may learn similar behaviors in shared regions to handle same the sub-task while overall policies still differing from each other due to other parts of own their main tasks. Hence, it makes training samples in shared regions become more diverse and possibly makes the learning process more stable and avoid local optimum that may happen in standard reinforcement learning methods because of insufficient samples or exploration.
- $W_i(a|s)$ is the important sampling weight which is the mixed action-samples distributions of task-policies k (including task-policy i) at state s where state s belongs to shared regions of task-policies k and task-policy i. π^{old} indicates the task-policy before update.
- Surrogate Objectives "clip": As sharing samples of task-policies in their shared regions helps task-policies solve their shared sub-problem more efficiently, that leads to the changes in state visitation density and action distribution, which can be harmful to the policy with respect to the main task. There are several works [Schulman *et al.*, 2015; Kakade and Langford, 2002] presenting theoretical constraint on policy-update that guarantees monotonic convergence. Later, Proximal Policy Optimization (PPO) algorithm [Schulman *et al.*, 2017] formalizes the constraint as a penalty in the objective function which gives the same or even better effect on policy improvement while having simple implementation. In this work, we borrow the concept of clipped objective function from PPO as a soft constraint to prevent excessively large policy update due to the difference between current task policy and the other policies.

Note that, there are several choices for Ψ. However, total reward of trajectory $\sum_{t=0}^{\infty} r_t$, total reward following action $\sum_{t'=t}^{\infty} r_{t'}$ or state-action function $Q^{\pi}(s_t, a_t)$ measure the success of action from current state to goal. They all are specific for a particular task while advantage function measures the success of current action compared to average at current state. Hence, it will avoid the conflicts caused by different reward schemes between tasks. Therefore, we choose advantage value function $A(s, a)$ for $\Psi_k(a_{k,t}|s_{k,t})$, so that $W_i(a_{k,t}|s_{k,t})\Psi_k(a_{k,t}|s_{k,t})\nabla_{\theta_i}\log\pi_{\theta_i}(a_{k,t}|s_{k,t})$ will increase $\pi_{\theta_i}(a_{k,t}|s_{k,t})$ when $A(s_{k,t}, a_{k,t}) > 0$ and vice versa. The value function V_{ϑ_i} used to estimate advantage function is updated as follows:

$$\text{minimize}_{\vartheta_i} \sum_{t=1}^{N} \left\| V_{\vartheta_i}(s_t) - \hat{V}_t \right\|^2 \quad (5)$$

where $\hat{V}_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$ is the discounted sum of rewards, and t indexes over all time steps in a batch of trajectories. This is sometimes called the Monte Carlo or TD(1) approach for estimating the value function [Sutton and Barto, 1998].

3.3 Learning Z_ϕ : A Confident Sharing-regions Feature-based Detection Agent

As introduced in the previous section, we want to learn $Z_\phi(s, i, j)$ agents that can detect the shared regions of two task-policies i and j. We consider Z_ϕ a neural agent parameterized by ϕ with input space S_z: (s, i, j) where $s \in$ state space S of N task-policies, i and j are task indexes and set of labels $L_z = \{1 : \text{share}, 0 : \text{non-share}\}$ tell if state s belongs to shared regions of two tasks i and j or not.

Remind that in their shared regions, task-policies may learn similar behaviors. Therefore, the task-specific reward signals (in advantage function value) should be similar when two policies perform the same action at a state. From this observation, we can analyze these reward signals to decide whether two policies should share the experience or not. Specifically, if reward signals feedbacked from the environment to the agent when two policies perform the same action at state s are both positive or negative, state s tends to belong to shared regions and vice versa.

We generate the samples to learn $Z_\phi(s,i,j)$ from samples of task-specific policies as follows: weight and label generated from collected state-action pairs (s,a):

$$W_z(s,i,j) = \omega_{weight}\begin{pmatrix}\mathbb{E}_{s_t=s,a_t=a}\left[A_{\vartheta_i}(s_t,a_t)\right],\\ \mathbb{E}_{s_t=s,a_t=a}\left[A_{\vartheta_j}(s_t,a_t)\right]\end{pmatrix} \quad (6)$$

$$L_z(s,i,j) = \omega_{label}\begin{pmatrix}\mathbb{E}_{s_t=s,a_t=a}\left[A_{\vartheta_i}(s_t,a_t)\right],\\ \mathbb{E}_{s_t=s,a_t=a}\left[A_{\vartheta_j}(s_t,a_t)\right]\end{pmatrix} \quad (7)$$

where two functions:

$$\omega_{weight}(a,b) = \begin{cases}\min(|a|,|b|) & \text{if} \quad a\times b\neq 0\\ p_z & \text{if} \quad a=0 \text{ or } b=0\end{cases} \quad (8)$$

and

$$\omega_{label}(a,b) = \begin{cases}1 & \text{if} \quad a\times b>0\\ 0 & \text{if} \quad a\times b\leqslant 0\end{cases} \quad (9)$$

There are states s which consistently appear in a task sample and absent from other ($a=0$ or $b=0$). We consider these states are specialized for a task and add a small penalty p_z to W_z to help Z_ϕ detect these specialized states. The weights of Z_ϕ are updated collaboratively to minimize the total loss $J_z(\phi)$ as follows:

$$J_z(\phi) = \mathbb{E}_{s,i,j}\left[-W_z(\cdot)\sum_{c\in\{0,1\}} L_z(c|\cdot)\log(Z_\phi(c|\cdot))\right] \quad (10)$$

In this work, we consider discrete environments and states and actions are compared directly. For continuous or large environments where experiences would be so sparse, it is potential to apply a confident metric to measure the similarity of states (acoss tasks) with respect to long-term behavior such as bisimulation metric [Ferns *et al.*, 2011].

3.4 Consistency between Sharing-experience Scheme and Z Agent

As discussed in the sharing-experience framework, policies in shared regions tend to be similar, which leads to the same observations and actions occurring when policies operate from a start point to the targeted goal. In this section, we will discuss the effect of $Z_\phi(s,i,j)$ and the sharing-experience framework to each other. Note that the sharing function $Z_\phi(s,i,j)$ is soft, so that agents can attempt to share learning information, even though they are not entirely confident that they are in a region that is shared. We divide sharing decisions of $z_\phi(s,i,j)$ into four cases (Positive: share, Negative: non-share):

- True-Positive (TP): policies i and j can exploit the experience of others to improve the learning process. It also results in observations and actions more likely to be similar after update and also gets the same responses from the environment. Then the confidence of Z_ϕ's prediction is reinforced.
- False-Positive (FP): policies i and j may learn from the wrong experiences. However, based on the constraint of sharing weight $S_{k,i}(a|s)$, we can prevent task-policies from large negative update. Besides, like TP case, it also results in observations and actions more likely to be similar the next time but gets different responses from the environment. Then the confidence of Z_ϕ's prediction is decreased.
- True-Negative (TN): No effect on the framework.
- False-Negative (FN): No effect on the framework.

It can be seen that sharing-experience framework and Z_ϕ mutually reinforce for TP and FP; meanwhile TN and FN can make Z_ϕ freeze because they have no effect on the framework. However, without Z_ϕ, task-policies still learn to have similar behaviors in shared regions. Hence, the classification of Z_ϕ tends to be TP and FP rather than TN and FN, and results in helping task-policies find their shared-regions and making them learn faster. The details of sharing-experience framework are presented in Algorithm. 1.

Algorithm 1 Sharing-experience framework with Z agent

```
Initialize policies π_{θ_i}, value functions V_{ϑ_i} and Z_φ
while not convergence do
  for i = 1 to N do
    Simulate policies π_{θ_i} until L steps
    Compute advantage estimates Â_{i,t} at all time step t
  end for
  for i = 1 to N do
    Optimize surrogate J_SEF wrt θ_i as eq. (3)
    Update value function parameters ϑ_i aseq. (5)
  end for
  for i = 1 to N do
    for j = 1 to N do
      if i ≠ j then
        Generate samples for Z_φ(s,i,j) from samples
        of task-policies as eqs. (6) and (7)
      end if
    end for
  end for
  Optimize J_z wrt φ as eq. (10)
end while
return solution
```

4 Experimental Evaluation

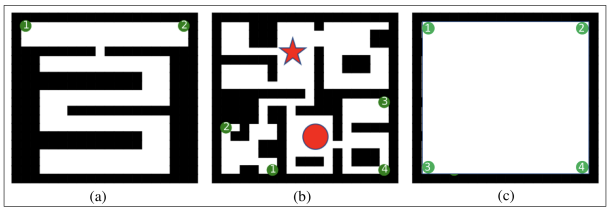

Figure 1: Experimental environments.

We focus our analysis on the scenario that agents perform in an environment with multiple tasks, each agent works on its own task. We subject our agents to three multi-task grid-world environments with various task-locations depicted in

Fig. 1 involving grid-world navigation where each agent is put in a random place at the beginning and the task is to find a way to the targeted position (the number presented task number). Agents receive a small negative reward for each step and a positive reward after reaching their own goal to encourage them to find the shortest trajectories to goal. The actions available to each agent are moving left, right, up, down, left-up, left-down, right-up and right-down.

We consider our SEF in two settings Oracle and Z, and determine the benefit of SEF, we compare it to one which learns task-policies independently and one which jointly trains all tasks simultaneously by a compound policy. We use basic advantage-actor-critic with separate policy network and value network as the base agent. Specifically, we compare these settings together:

- **Oracle-sharing:** agents are provided information about shared regions between tasks, this is the upper bound of our algorithm. $Z_\phi(s, i, j)$ only returns $[0, 1]$ or $[1, 0]$ in this settings.
- **Z-sharing:** agents learn themselves where they should share the experience by Z_ϕ. For this setting, we add a small sharing confident bound Z_c to Z_ϕ to guarantee shared regions prediction: $Z_\phi(s, i, j) = [\max(Z_\phi(1|s, i, j) - Z_c, 0), \min(Z_\phi(0|s, i, j) + Z_c, 1)]$.
- **Baseline-1 (B-1):** training task-policies independently.
- **Baseline-2 (B-2):** training all tasks simultaneously with a compound policy.

In the experiments, the agents' hyper-parameters and network structure are the same for all settings. The details of hyper-parameters and network structures are presented in Table 1 and Table 2.

Hyper-parameters	**Setting**
State dimension	one-hot representation
Action dimension	8
Discounted factor	0.99
Number of Iterations	1000
Number of rollouts each Inter.	8, 16, 24
Rollout length: L	50
Learning rate actor-critic	0.005
Learning rate Z	0.025
ϵ	0.2
Z_c	0.1
p_z	0.001
Optimizer	RMSProp

Table 1: Hyper-parameter settings

4.1 Experimental Details and Discussion

Multiple Goals Well-Gated Grid-World

In this environment, the experience before the gating can be shared, but the experience after the gating cannot. From that, we can observe the ability to find shared regions of Z_ϕ.

In the first experiment, there are two tasks with the goals located either to the left or to the right at the top of the environment. The Z_ϕ agent of SEF has to detect the shared regions between two tasks and uses this information to make task-policies share their experience with each other.

Network	**In**	**F.C.1**	**F.C.2**	**Out**
(B-1, SEF) Policy	324	256	256	8
(B-1, SEF) Value	324	256	256	1
(B-2) Policy	324+N	256	256	8
(B-2) Value	324+N	256	256	1
Z net	324	256	256	2
N		task index in one-hot		

Table 2: Network structures

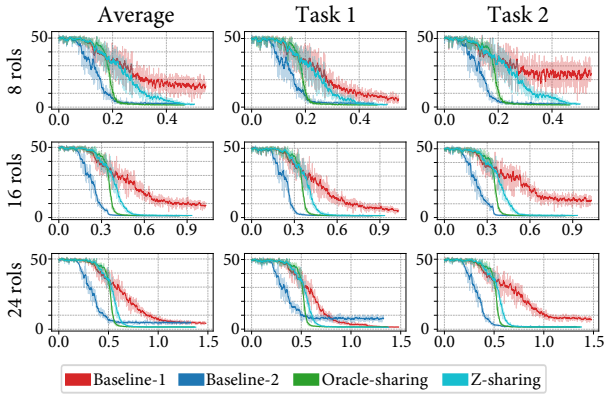

Figure 2: The average performance (redundant steps/million training step) across all the tasks and the performance of each task specific policy in scenario.1, respectively.

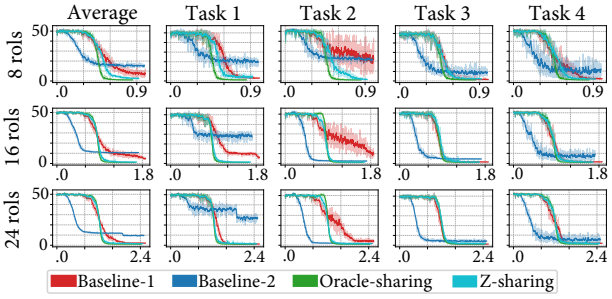

Figure 3: The average performance (redundant steps/million training step) across all the tasks and the performance of each task specific policy in scenario.2, respectively.

For second experiment, we design the environment as a house with five rooms and much more complicated than environment 1. There are four tasks, which are divided into two groups located in two rooms, one at the top left and one at the top right. For this scenario, the sharing context of task-specific policies is different for agents' locations. For example, as shown in Fig. 1 of environment 2, when agents stay in the circle point, the task-1 policy and task-2 policy should share experience with each other but should not share with task-3 and task-4 policies. On the contrary, when agents stay in start point, all four task policies should mutually share with each other.

As discussed in Sec 3.2, our SEF enhance data-efficiency and is able to handle local optimums caused by insufficient samples or exploration of the standard method. To analyzing this assumption, we conduct experiments with different numbers of rollouts each iteration and observe the learning curves of SEF and two baselines.

Results of the learning performance of SEF and two baselines with different numbers of rollouts are shown in Fig. 2 and Fig. 3. Baseline-1 learns task-specific policies independently, so this approach can not exploit experience between tasks and take long time to convergence. Learning process of the Baseline-2 may be more efficient through average updates across related tasks. As illustrated in Fig. 2, when two tasks have little conflict, Baseline-2 work very well. However, as environment and tasks get complex, such policy may get biased towards a task due to the gradient updates from tasks can negate each other, making the learning process unstable. The result is some tasks stuck in local optimums (Fig. 3). In contrast, SEF not only improves the convergence rate in comparison to Baseline-1 setting but also helps agent avoid local optimums in both oracle-sharing and z-sharing settings in comparison to two baselines. Furthermore, the learning curves of the agents in SEF are also more stable than those of independent task-policies training framework's agents, especially for SEF agents trained with a low number of rollouts each epoch. However, the dominance of out SEF decreases as the number of rollouts increases. The reason is that the variance of MCMC can decrease when the number of samples increases. The performance of Baseline-2 is sometimes better than SEF's for some tasks, and it is because the updated efficiency may be improved by transferring knowledge across related task. However, gradients from different tasks can interfere negatively, making some tasks stuck in local optimums. As illustrated in Fig. 2, when this environment is quite simple and the task-polices are not too conflicted, the performance of Baseline-2 almost reaches SEF's oracle settings' performance. However, in Fig. 3, where the environment is more complex, there are some tasks stuck in local optimums.

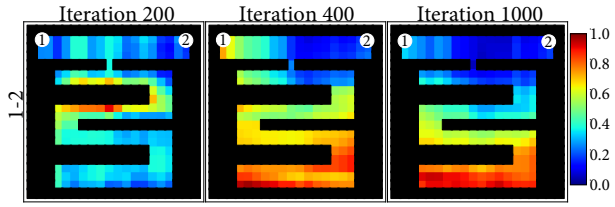

Figure 4: Sharing confidence of Z_ϕ between tasks 1-2 of sharing-scheme in scenario.1, in 16 rollouts per iteration setting.

Besides, we also obverse that the learning curves of Z-sharing almost reaches the performance of Oracle-sharing setting. That means Z_ϕ agent does detect the sharing-regions between tasks well. This is also proved by the sharing confidence between tasks of Z_ϕ over iteration provided by Fig. 4 and Fig. 5. As shown in Fig. 5, the shared regions of task 1 and task 2 located at the same room are larger than those of task 1 and task 4 located at different rooms.

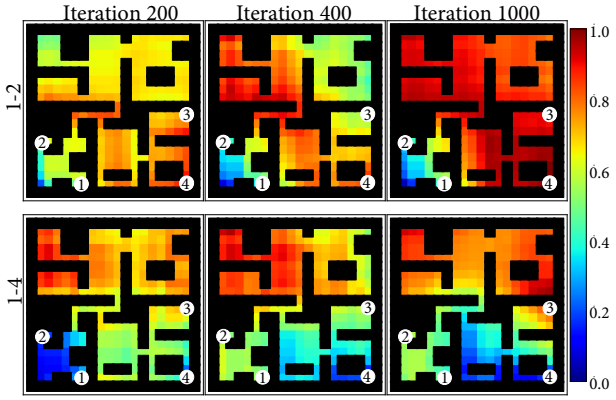

Figure 5: Sharing confidence of Z_ϕ between tasks 1-2 and tasks 1-4 of sharing-scheme in scenario.2, 16 rollouts per iteration setting.

Multiple Goals Grid-World without Wall

In this environment, there are fours goals located in corners of a square grid-world without wall, so the explicit partitioning for shared regions could be difficult, the true probability of being in a shared region changes continuously. As the sharing confident map illustrated in Fig.6, our framework can make the appropriate trade-off between using shared information.

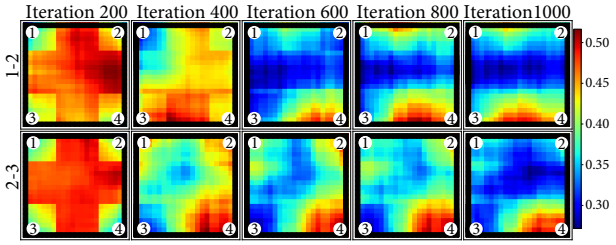

Figure 6: Sharing confidence of Z_ϕ between tasks 1-2 and tasks 2-3 in grid-world without wal, 16 rollouts per iteration setting.

5 Conclusion

Our work illustrates that latent variable representing shared portions across various tasks enhance training agents in multi-task scenarios. In the paper, we have proposed sharing experience reinforcement learning framework, a framework that:(1) identifying shared portions among tasks through task-specific rewards from the environment, (2) proposing a soft sharing experience framework based on shared portions that enhance data-efficiency over task-policies to make learn process more stable and robust. Our experiment's results show that sharing experience reinforcement learning framework substantially outperforms standard RL algorithm. For future work, we focus on applying our approach on continuous space and, as our shared regions can be explicitly represented in state space, it is promising to extracted shared regions as based skills to accomplish more complex tasks.

Acknowledgements

This research is based upon work supported by the US AOARD/AFOSR under award number FA2386-17-1-4053.

References

[Andreas *et al.*, 2016] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. *CoRR*, abs/1611.01796, 2016.

[Borsa *et al.*, 2016] Diana Borsa, Thore Graepel, and John Shawe-Taylor. Learning shared representations in multi-task reinforcement learning. *CoRR*, abs/1603.02041, 2016.

[Devin *et al.*, 2017] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2169–2176, May 2017.

[Ferns *et al.*, 2011] N. Ferns, P. Panangaden, and D. Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011.

[Graves *et al.*, 2013] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[Grounds and Kudenko, 2008] M. Grounds and D. Kudenko. Parallel reinforcement learning with linear function approximation. In *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, pages 60–74. Springer, 2008.

[Gupta *et al.*, 2017] Abhishek Gupta, Coline Devin, Yuxuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *CoRR*, abs/1703.02949, 2017.

[Hausman *et al.*, 2018] K. Hausman, J.T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[Kakade and Langford, 2002] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274, 2002.

[Konidaris *et al.*, 2011] G. Konidaris, S. Osentoski, and P. Thomas. Value function approximation in reinforcement learning using the fourier basis. In *AAAI*, volume 6, page 7, 2011.

[Krizhevsky *et al.*, 2012] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LeCun *et al.*, 2015] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[Lillicrap *et al.*, 2015] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[Mnih *et al.*, 2015] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, and S. Petersen. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[Mnih *et al.*, 2016] V. Mnih, A.P. Badia, M. Mirza, Lillicrap T. Graves, A., T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[Parisotto *et al.*, 2015] E. Parisotto, J.L. Ba, and R. Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *CoRR*, abs/1511.06342, 2015.

[Rusu *et al.*, 2015] A.A. Rusu, S.G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *CoRR*, abs/1511.06295, 2015.

[Rusu *et al.*, 2016] A.A. Rusu, N.C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust region policy optimization. In *Icml*, volume 37, pages 1889–1897, 2015.

[Schulman *et al.*, 2017] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[Sutton and S, 1988] Sutton and Richard S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[Teh *et al.*, 2017] Y. Teh, V. Bapst, W.M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. *CoRR*, abs/1707.04175, 2017.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[Wilson *et al.*, 2007] A. Wilson, A. Fern, S. Ray, and P. Tadepalli. Multi-task reinforcement learning: A hierarchical bayesian approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 1015–1022, New York, NY, USA, 2007. ACM.

[Yang *et al.*, 2017] Z. Yang, K.E. Merrick, H.A. Abbass, and L. Jin. Multi-task deep reinforcement learning for continuous action control. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3301–3307, 2017.

[Zhang *et al.*, 2017] J. Zhang, J.T. Springenberg, J. Boedecker, and W. Burgard. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2371–2378, Sept 2017.