MULTI-TASK REINFORCEMENT LEARNING WITH TASK REPRESENTATION METHOD

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-task reinforcement learning (RL) algorithms can train agents to acquire generalized skills across various tasks. However, jointly learning with multiple tasks can induce negative transfer between different tasks, resulting in unstable training. In this paper, we newly propose a task representation method that prevents negative transfer in policy learning. The proposed method for multi-task RL adopts a task embedding network in addition to a policy network, where the policy network takes the output of the task embedding network and states as inputs. Furthermore, we propose a measure of negative transfer and design an overall update method that can minimize the suggested measure. In addition, we raise an issue of the negative effect on soft Q-function learning resulting in unstable Q learning and introduce the clipping method to reduce this issue. The proposed multi-task algorithm is evaluated on various robotics manipulation tasks. Numerical results show that the proposed multi-task RL algorithm effectively minimizes negative transfer and achieves better performance than previous state-of-the-art multi-task RL algorithms.

1 INTRODUCTION

Reinforcement learning (RL) with deep neural networks has been applied successfully to various fields, e.g., playing Atari games from raw pixel images (Mnih et al., 2015), the game of Go (Silver et al., 2016), control of locomotion skills (Schulman et al., 2015; Lillicrap et al., 2015; Schulman et al., 2017; Haarnoja et al., 2018b; Fujimoto et al., 2018), etc. Despite such success of deep RL, most deep RL algorithms solve each task independently so that they suffer from the lack of sample efficiency on complex tasks. Learning multiple tasks with individual policies requires a large amount of memory and samples, whereas training a single policy network that generalizes across a given set of tasks is challenging in RL. In particular, generalized skills across diverse tasks are necessary to apply RL algorithms in real robot control. Multi-task learning (Caruana, 1997) is one approach to this problem. In multi-task RL, the agent needs to train a policy that can be generalized across diverse sets of tasks. The agent can train a policy parameterized by a neural network with multiple tasks, improving sample efficiency by sharing and reusing the parameters across different tasks.

However, multi-task RL has a challenging problem called negative transfer in which the training of some tasks can negatively affect other tasks, resulting in instability in training (Sun et al., 2019). For example, Fig. 1 shows that the gradients of task 1 and tasks 2 and3 do not align, so the gradient direction of task 1 harms the learning of tasks 2 of 3. Note that as the number of tasks increases, the negative transfer occurs more often. To tackle this issue, the policy with multiple modules (Haarnoja et al., 2018a; Andreas et al., 2017; Yang et al., 2020) was introduced to prevent the interference between tasks using the different module for each task. In particular, the gradient surgery method proposed in (Yu et al., 2020a) keeps off negative transfer by aligning the gradient direction of each task using the projection method. In practice, however, the gradient of each task contains large noisy component because it is computed by sampling a mini-batch from replay buffer, resulting in an inaccurate projection of gradient.

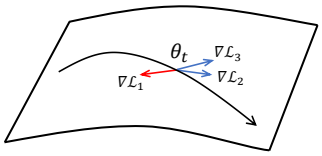


Figure 1: Negative transfer between Task 1 and Tasks 2/3

In this paper, we focus on preventing negative transfer on both policy π and its action value function Q^π. We propose a task representation method which effectively restrains the negative transfer in policy learning. This approach consists of a task embedding network and a policy network and a learning algorithm to minimize interference between tasks. The task embedding network receives its input from the task ID designed as a one-hot encoding vector and the current state, and outputs the representation of task information at the current time. The policy network takes the current state and the representation from the task embedding network as input, and outputs an action at the current time. In the learning process, we measure the amount of negative transfer according to the update of policy for each task, and train the task embedding network to minimize the measurement through a gradient-based meta-learning approach (Finn et al., 2017; Ji et al., 2020). Then, we analyze the interference on Q function between different tasks, and introduce a simple clipping method on Q function, allowing stable Q learning. The detail will be explained in the following sections. We evaluate the proposed method in Meta-World (Yu et al., 2020b), which contains 50 manipulation tasks using the robotic arm, and achieve better performance than previous state-of-the-art multi-task algorithms.

2 Background

2.1 Setup

We consider RL composed of an environment E and an agent, and assume that the system is modeled as a discrete-time finite-horizon Markov decision process (MDP) denoted by $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho, \gamma, H)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ is the transition probability, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$ is the reward function, $\rho : \mathcal{S} \to \mathbb{R}_+$ is the initial state distribution, $\gamma \in [0, 1)$ is the discount factor, and H is the time horizon. We assume that the environment is fully-observable so that the environment state s_t is available to the agent. At each time step t, the agent observes a state $s_t \in \mathcal{S}$ from the environment E and executes an action $a_t \in \mathcal{A}$ based on its policy $\pi(a_t|s_t)$. Then, the environment gives the agent a reward r_t according to the reward function $r_t = r(s_t, a_t)$ and makes a transition to a next state s_{t+1} according to the transition probability $\mathcal{P}(s_{t+1}|s_t, a_t)$. The goal of basic RL is to maximize the expected discounted accumulated rewards $J(\pi) = \mathbb{E}_\pi\Big[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\Big]$. We use parameterized policy $\pi_\theta(a_t|s_t)$ which is trained to maximize $J(\pi_\theta)$.

2.2 Soft Actor-Critic

Soft Actor-Critic (SAC) (Haarnoja et al., 2018b) is an off-policy RL algorithm, where the action from the actor is encouraged to be sampled uniformly by increasing the entropy of action. The policy objective function of SAC is given by $J_{SAC}(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t (r_t + \beta\mathcal{H}(\pi(\cdot|s_t)))]$ where $\mathcal{H}$ is the entropy function and $\beta > 0$ is the weighting for the entropy term. To optimize $J_{SAC}(\pi)$, SAC optimizes alternately the parameterized Soft Q-function Q_ψ and the policy π_θ. The policy loss function and soft Q-function loss are given by

$$J^\pi(\theta) = \mathbb{E}_{s\sim\rho}\big[\mathbb{E}_{a\sim\pi}[\beta \log \pi_\theta(a|s) - Q_\psi(s, a)]\big], \tag{1}$$

$$J^Q(\psi) = \mathbb{E}_{s\sim\rho, a\sim\pi}\Big[\frac{1}{2}\big(Q_\psi(s, a) - (r(s, a) + \gamma\mathbb{E}_{s',a'}[Q_{\bar{\psi}}(s', a') - \beta \log \pi_\theta(a'|s')])\big)^2\Big], \tag{2}$$

where $\bar{\psi}$ is a parameter for target Q-value network. The coefficient β is trained to maintain the entropy $\mathcal{H}(\pi(\cdot|s_t))$ to a target entropy $\bar{H}$ (Haarnoja et al., 2018c) with

$$J(\beta) = \mathbb{E}_{a\sim\pi}[-\beta \log \pi(a|s) - \beta\bar{H}]. \tag{3}$$

2.3 Multi-task Reinforcement Learning

Under the setup of multi-task RL, we consider a task set $\mathcal{C} = \{\mathcal{T}_i\}_{i=1}^N$ and assume a uniform distribution $p(\mathcal{T})$ over the task set $\mathcal{C}$. Here, each task $\mathcal{T}_i$ has a different MDP $M_i = (\mathcal{S}, \mathcal{A}, \mathcal{P}_i, r_i, \rho_i, \gamma, H)$, and captures a situation where the reward function r_i and the transition probability $\mathcal{P}_i$ are different across the tasks. Each task $\mathcal{T}$ drawn from the distribution $p(\mathcal{T})$ has a maximization objective given by the expected discounted accumulated rewards, $J(\pi, \mathcal{T})$. The goal of Multi-task RL is to learn a policy that maximizes the overall expected returns over the tasks $\mathbb{E}_{\tau\sim p(\mathcal{T})}[J(\pi, \mathcal{T})]$

In this paper, we train a generalized policy $\pi(a|s, z)$ across the tasks with the SAC algorithm as the background algorithm, where z represents an encoding of the task ID, i.e., one-hot task identification encoding. The policy objective of multi-task RL is then given by $\mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}[J_{SAC}(\pi, \mathcal{T})]$.

We assume that the agent has an individual soft Q-function $Q_i^\pi(s, a)$ for each task $\mathcal{T}_i$, where each soft Q-function is trained by the loss (2). The coefficient of entropy β_i exists for each task $\mathcal{T}_i$ in the multi-task RL setting, and each β_i is optimized by $J(\beta_i) = \mathbb{E}_{a\sim\pi,\mathcal{T}_i}[-\beta_i \log \pi(a|s, z) - \beta_i \bar{H}]$.

3 Multi-Task Reinforcement Learning with Task Embedding Network

In multi-task learning, we wish that the knowledge that is acquired from learning each task can be exploited and helps learn other tasks better. During multi-task learning, however, the training of one task can negatively affect the training of others, resulting in negative transfer which induces instability in learning. In this section, we investigate how the negative transfer affects other tasks' soft Q-function Q_i^π learning, and introduce a simple clipping technique to prevent from harming Q_i^π. Then, we propose a task embedding network for multi-task RL, which provides task representation to policy and prevents the negative transfer effect on policy training. From the proposed task embedding network, we introduce a measure of negative transfer for each task, which is used to reduce interference with other tasks in policy training. Finally, we propose an additional experience replay per task, which stores the top K trajectories on return for the task embedding network training.

3.1 Policy Network with Task Embedding Network

In the proposed method, we have a generalized policy $\pi_\theta(a|s, z)$ parameterized by θ, and a task embedding network $E_\phi(z|s, z_\mathcal{T})$ parameterized by ϕ for each task $\mathcal{T}$. Here, the proposed networks are shared across all tasks. Note that $E_\phi(\cdot|s, z_\mathcal{T})$ is a distribution over the representation space Z. At each time, the task embedding network takes the input of the current state s_t and the embedding of the task ID $z_\mathcal{T}$, and outputs the representation $z \sim E_\phi(z|s, z_\mathcal{T})$, where $z_\mathcal{T}$ is one-hot task identification encoding. The policy $\pi_\theta(a|s, z)$ is learned to achieve the goal of multi-task RL by taking the representation z as an additional input in addition to the current state s_t.

3.2 Measure of Negative Transfer

For each task $\mathcal{T}_i$, we have the policy objective as $J_{SAC}(\pi_\theta, E_\phi, \mathcal{T}_i)$, and the policy parameter θ should be trained to maximize $\mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}[J_{SAC}(\pi_\theta, E_\phi, \mathcal{T})]$ by gradient ascent. However, there can exist some task $\mathcal{T}_k \in \mathcal{C}$ whose gradient direction for its objective harms the objectives of other tasks, and this makes it challenging to achieve the goal of the multi-task RL. Therefore, to tackle this problem, we investigate when negative transfer can occur. Note that for notational simplicity, denote $J_{SAC}(\pi_\theta, E_\phi, \mathcal{T})$ as $J_{SAC}(\theta, \phi, \mathcal{T})$.

Negative Transfer: Consider a task objective set $J_{set} = \{J_{SAC}(\theta, \phi, \mathcal{T}_i)\}_{i=1}^N$. From the policy objective $J_{SAC}(\theta, \phi, \mathcal{T}_i)$ of each task $\mathcal{T}_i$, we update the policy parameter θ to maximize the policy objective by gradient-ascent. Let this updated policy parameter be θ^i, i.e.,

$$\theta^i = \theta + \alpha \nabla_\theta J_{SAC}(\theta, \phi, \mathcal{T}_i), \tag{4}$$

where $\alpha > 0$ is the gradient step size. From the updated policy parameter θ^i, we renew the task objective set J_{set} to J_{set}^i:

$$J_{set}^i = \{J_{SAC}(\theta^i, \phi, \mathcal{T}_j)\}_{j=1}^N. \tag{5}$$

Assuming that a task $\mathcal{T}_k$ causes many negative transfers to other tasks, the task $\mathcal{T}_k$ increments the number of tasks whose policy objective is lower than before the parameter was updated to θ^k. Let us denote the number of such tasks as $\mu(\mathcal{T}_k)$, i.e.,

$$\mu(\mathcal{T}_k) = \left|\{i \mid J_{SAC}(\theta^k, \phi, \mathcal{T}_i) < J_{SAC}(\theta, \phi, \mathcal{T}_i)\}\right| \tag{6}$$

Measurement of Negative Transfer: $\mu(\mathcal{T}_k)$ in Equation 6 indicates how many tasks are negatively affected by updating the parameter θ to θ^k. Hence, $\mu(\mathcal{T}_k)$ can be considered as a measure of negative transfer on task $\mathcal{T}_k$. So, we define $\mu(\mathcal{T}_k)$ as the measure of negative transfer caused by task $\mathcal{T}_k$. Note that when $\mu(\mathcal{T}_k)$ is large, task $\mathcal{T}_k$ induces large negative transfer.

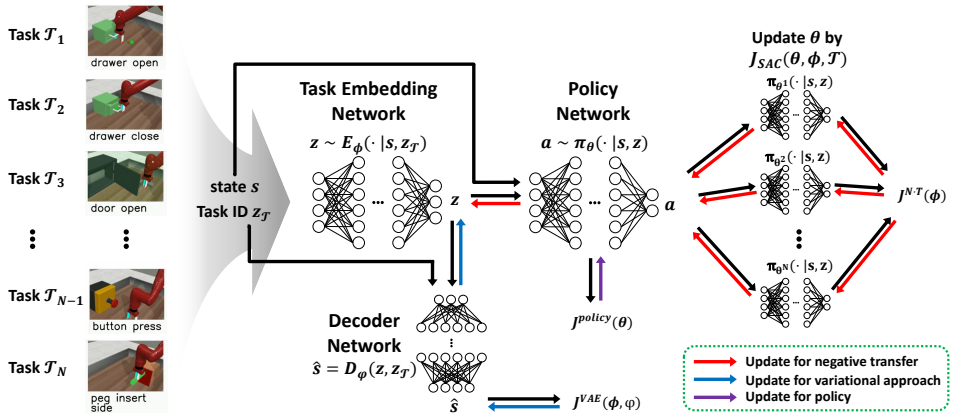


Figure 2: Architecture for the proposed method. Our architecture consists of three network: (1) task embedding network E_ϕ (2) policy network π_θ (3) decoder network D_φ.

3.3 Training for Policy Network

The goal of training policy π_θ is to maximize the average expected return over the task set C, i.e., $\mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}[J_{SAC}(\theta, \phi, \mathcal{T})]$. Based on the measure of negative transfer defined above, we determine the set of tasks benign to other tasks in the middle of learning, and propose an update rule to train the policy with learning stability.

Task Selection: We choose a benign set C_s of tasks from the task set C. For this, we order the negative transfer measure values $\mu(\mathcal{T}_i)$, $i = 1, \cdots, N$ and then select the tasks with the smallest M values. That is, $C_s := \{\mathcal{T}_j \mid \mu(\mathcal{T}_j) \leq \mu(\mathcal{T}_{\kappa(M)})\}$, where $\kappa(i)$ is defined as

$$\kappa(i) = \underset{\substack{k \in \{1, \cdots N\} \\ k \notin \{\kappa(1), \cdots \kappa(i-1)\}}}{\arg\min} \quad \mu(\mathcal{T}_k) \quad \text{for} \quad i = 1, \cdots M. \tag{7}$$

Then, instead of using the policy loss equally weighted over the task set C, we give the tasks in C_s more weight to reduce negative transfer on the policy learning. The weight w_i for task $\mathcal{T}_i$ is proportional to the exponential of $N - \mu(\mathcal{T}_i)$ as

$$w_i = \frac{\exp\left(N - \mu(\mathcal{T}_i)\right)}{\sum_{j=1}^{N} \exp\left(N - \mu(\mathcal{T}_i)\right)}, \tag{8}$$

where N is the number of tasks. Thus, we consider the following objective function for the policy parameters θ:

$$J^{\text{policy}}(\theta) = \sum_{i=1}^{N} w_i \, J_{SAC}(\theta, \phi, \mathcal{T}_i). \tag{9}$$

3.4 Training for Task Embedding Network

In order to prevent negative transfer, the task embedding network E_ϕ should be trained to decrease $\mu(\mathcal{T}_k)$ for all tasks $\mathcal{T}_k \in C$. However, it is difficult to decrease $\mu(\mathcal{T}_k)$ directly because $\mu(\mathcal{T}_k)$ is not differentiable with respect to the parameter ϕ. Thus, we consider an alternative cost function for ϕ as maximizing the average over the task objective set in Equation 5 for each task $\mathcal{T}_k$:

$$J^{N \cdot T}(\phi, \mathcal{T}_k) = \frac{1}{N} \sum_{i=1}^{N} \Big[J_{SAC}(\theta^k, \phi, \mathcal{T}_j) \Big]. \tag{10}$$

Since the update rule in Equation 4 also depends on the parameter ϕ, we rewrite the Equation 10 as

$$J^{N \cdot T}(\phi, \mathcal{T}_k) = \frac{1}{N} \sum_{i=1}^{N} \Big[J_{SAC}(\theta + \alpha \nabla_\theta J_{SAC}(\theta, \phi, \mathcal{T}_k), \phi, \mathcal{T}_j) \Big]. \tag{11}$$

Thus, optimizing the objective in Equation 11 can be seen as training the task embedding network E_ϕ not to perform well for a given policy π_θ but to perform well after updating θ to θ^k, similar to gradient-based meta-learning approaches (Finn et al., 2017; Ji et al., 2020).

For training of the embedding network, rather than considering all tasks in C, we focus on the tasks in the set $C \setminus C_s$, which is the set of harmful tasks in training policy π_θ, and mitigate the negative effect by each task $\mathcal{T} \in C \setminus C_s$. Thus, the final objective function for parameter ϕ to suppress negative transfer on policy learning is written as

$$\widehat{J}^{N \cdot T}(\phi) = \frac{1}{|C \setminus C_s|} \sum_{\mathcal{T} \in C \setminus C_s} J^{N \cdot T}(\phi, \mathcal{T}). \tag{12}$$

Variational Approach to Task Embedding Network: The representation $z \sim E_\phi(\cdot|s, z_\mathcal{T})$ should capture the sufficient feature of each task and the current state so that the policy can generalize over the task set C by taking the representation z as an additional input. In order to capture relevant features of state and task, we consider an additional objective through a variational approach (Kingma & Welling, 2013).

Let the decoder network $D_\varphi(z, z_\mathcal{T})$ be parametrized by φ, which takes the representation z from the embedding network E_ϕ and the embedding of the task ID $z_\mathcal{T}$ as input. The goal of the decoder network is to reconstruct the state s from $z \sim E_\phi(\cdot|s, z_\mathcal{T})$ and $z_\mathcal{T}$ so that the representation z becomes a sufficient feature for current state s and task information $\mathcal{T}$. Thus, we consider an additional objective through variational approach in addition to the negative transfer loss:

$$J^{VAE}(\phi, \varphi) = \mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})} \Big[\mathbb{E}_{z \sim E_\phi, s \sim \mathcal{T}} \big[|D_\varphi(s, z_\mathcal{T}) - s|^2 + D_{KL}(E_\phi(\cdot|s, z_\mathcal{T}), p(z)) \big] \Big], \tag{13}$$

where $p(z)$ is the standard normal distribution. Then, we obtain the final objective for the task embedding network and the decoder network as follows:

$$J^{\text{rep}}(\phi, \varphi) = \widehat{J}^{N \cdot T}(\phi) - c_{vae} J^{VAE}(\phi, \varphi), \tag{14}$$

where $c_{vae} > 0$ is the weighting coefficient between the two terms.

3.5 Negative Effects on Soft Q-function

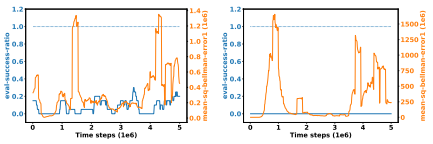
In practice, we consider an individual soft Q-function $Q_{\psi^i}(s, a)$ parameterized by ψ^i for each task $\mathcal{T}_i$ so that the parameters $\{\psi^i\}_{i=1}^N$ are optimized individually. For a given state, action, reward, and next state pair $d = (s, a, r, s')$, we can write the soft Bellman error of Q-function as

$$J^Q(d, \psi^i) = \frac{1}{2} \Big(Q_{\psi^i}(s, a) - (r + \gamma \mathbb{E}_{a' \sim \pi, z \sim E_\phi} [Q_{\bar{\psi}^i}(s', a') - \beta \log \pi_\theta(a'|s', z)]) \Big)^2, \tag{15}$$

where $\bar{\psi}^i$ is the parameter of the target Q-function network for each task $\mathcal{T}_i$.

If the current policy parameter θ is significantly affected from other tasks, the Q-value $Q(s', a')$ in Equation 15 has a poor estimate because the distribution of the replay buffer may not correspond with the distribution under the current policy, which results in unstable learning of soft Q-function (Fujimoto et al., 2019).

To see the negative effect on soft Q-value, we experiment a multi-task SAC algorithm (Yu et al., 2020b) with a shared policy and individual soft Q-function on MT10 benchmarks in the Meta-World (Yu et al., 2020b), and investigate the soft Bellman error for each task. Fig. 3 shows the success ratio and soft Bellman error of two specific tasks (Reach-v2 and Pick-place-v2) in MT10, where the orange curve shows the average of the Bellman error $J^Q(d, \psi^i)$, and the blue curve shows the average of success ratio of the given task. It is seen that the values of Bellman error become very large after a certain time step, significantly reducing the success ratio of each task. Note that the scale of Bellman error is 1e6.



(a) Reach-v2 (b) Pick-Place-v2

Figure 3: Soft Bellman error and success ratio graph: (a) Reach-v2 and (b) Pick-place-v2

Clipping Q Loss: In order to solve the instability of individual soft Q-function learning, we propose a simple technique that clips the soft Bellman error (15) for a given pair (s, a, s') not to be so large. Thus, the individual Q-function loss with clipping is given by

$$J^Q_{CLIP}(\psi^i) = \mathbb{E}_{d=(s,a,r,s')\sim D}\Big[\text{clip}\big(J^Q(d, \psi^i), 0, V_{\text{clip}}\big)\Big], \tag{16}$$

where D is mini-batch and V_{clip} is a maximum value of that does not clip.

3.6 Top-K Experience Replay

The policy objective J_{SAC} is obtained by sampling a mini-batch from the experience replay so that the selected task set C_s may change largely depending on the sampled mini-batch. Thus, instead of selecting the task set C_s for every time step, we select the task set C_s using the task selection method for every H steps, where H is the horizon length of environments. In the original experience replay buffer, there are many trajectories whose return may be high or low, and these diverse samples can make the uncertainty on the task selection method. Since the chosen task set C_s plays an important role in preventing negative transfer in our method, the proposed task selection method should be more reliable. In order to reduce the uncertainty in the task selection method, we build another buffer that stores K trajectories with high returns from the original buffer, and train the task embedding network E_ϕ by sampling mini-batchs from this replay buffer storing high-return trajectories. The overall architecture of the proposed method is illustrated in Fig. 2 and is summarized in Algorithm 1

4 Experiment

In this section, we describe the environment setting, baselines, and implementation details.

4.1 Environment

Each environment used in our experiment has a distinct task set $\mathcal{C} = \{\mathcal{T}_1, \ldots, \mathcal{T}_N\}$, where N is the number of tasks. Each task's reward function and transition probability are different, but the remaining setup such as the state and action spaces is identical. In order to test how well the proposed method can perform on multiple complex tasks, we tested our approach with the environment proposed in Meta-World (Yu et al., 2020b), which has 50 distinct robotic control tasks with a sawyer arm in the MuJoCo environment (Todorov et al., 2012). Our experiments use two multi-task benchmarks, MT10 (Yu et al., 2020b) and MT20 with 10 and 20 manipulation tasks, respectively, from the Meta-World environment. The task sets of MT10 and MT20 are given in Appendix B.

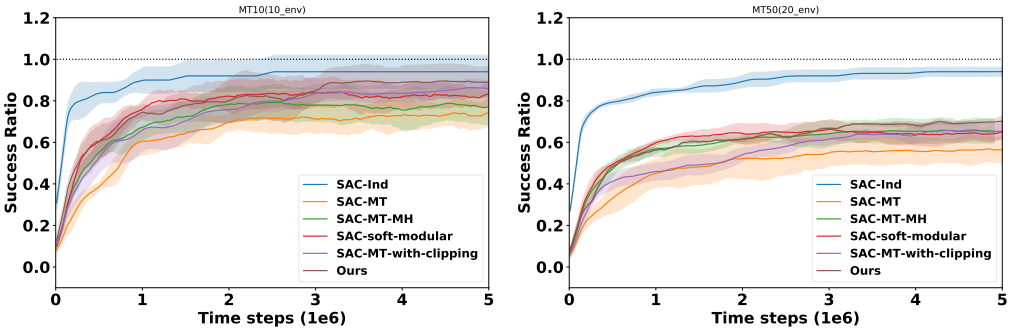
4.2 Baselines

We compared the proposed method with the baseline methods: 1) SAC (Haarnoja et al., 2018b) with a single-task (SAC-Ind): An individual policy for each task with the SAC algorithm. 2) SAC with multi-task (SAC-MT): A shared policy with a one-hot task identification encoding and current state as input. 3) SAC with Multi-task Multi-head (SAC-MT-MH) (Yu et al., 2020b): It is similar to SAC with multi-task but has an independent final layer in the policy network for each task (multi-head). 4) SAC with soft modularization (SAC-soft-modular) (Yang et al., 2020): Policy with multiple modules with soft modularization technique that gives routing strategy for each task. 5) SAC with clipping in multi-task setting (SAC-MT-with-clipping): SAC-MT with our proposed clipping method on soft Q-function learning.

5 Results

In this section, we evaluate the proposed multi-task SAC with a task embedding network in various robotics manipulation tasks. We first provide a performance comparison between the proposed method and previous multi-task RL methods on several benchmarks and then give an ablation study on components constituting the proposed algorithm.

5.1 Results on multiple tasks



(a) MT10 result

(b) MT20 result

Figure 4: Training curves for the average success ratio comparing the baseline algorithms and our approaches: (a) MT10 benchmark (b) MT20 benchmark

We first examined the performance of our approach compared with baseline algorithms on MT10 and MT20 benchmarks. Fig. 4 shows the training curve of the average success ratio, and Table 1 shows the final average success ratio for each algorithm. The success ratio is obtained by the average of all tasks' success ratios for given task set, where the success ratio of each task is well defined (Yu et al., 2020b). We ran the experiment 5 times for each algorithm with different seeds and plotted the mean success ratio with standard deviation per every 200 episodes. We trained each algorithm with 5 million samples per task on the MT10 and MT20 task sets.

Algorithm	MT10	MT20
SAC-Ind	**94.0%**	**94.1%**
SAC-MT	74.3%	56.2%
SAC-MT-MH	76.9%	65.3%
SAC-soft-modular	83.3%	65.0%
SAC-MT-with-clip	85.9%	65.5%
Ours	**89.3%**	**70.1%**

Table 1: Final average success ratio of MT10 and MT20 tasks

Results on MT10. As seen in Table 1, our approach's final success ratio is not only close to single task SAC (upper bound) but also 6% better than the best multi-task baseline algorithm SAC-soft-modular (Yang et al., 2020). Furthermore, SAC-MT combined with our Q clipping method also outperforms other baseline algorithms but has lower performance than our algorithm.

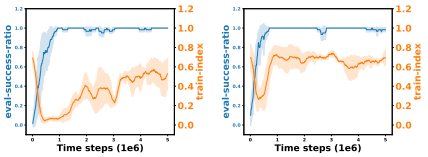
Results on MT20. As the number of tasks increases, the multi-task RL problem becomes more challenging since more negative transfer can occur. Nevertheless, as seen in Table 1, our approach achieves the best final success ratio compared with other multi-task baselines, where the proposed method has 4.8% better success ratio than the best multi-task baseline algorithm SAC-MT-MH (Yu et al., 2020b). Furthermore, the SAC-MT combined with our Q clipping method is also similar to other baseline algorithms but has lower performance than our algorithm.

The training curves for all tasks in the benchmarks are shown in Appendix C.

5.2 Ablation Studies

Effects on Method for Preventing Negative Transfer

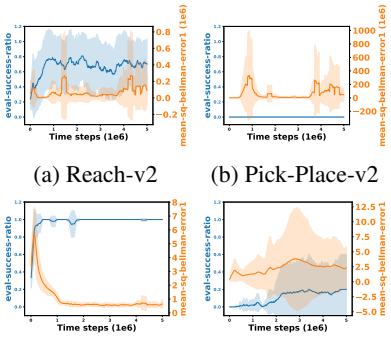
In our method, the measure of negative transfer for a given task is important. In order to see how the measure changes during the learning, we investigated the relationship between the success ratio and the train frequency for a given task. Here, the train frequency is the rate at which the policy is trained on a given task over a period of 20 episodes, inversely proportional to the measure of negative transfer from the definition of the task set C_s in Equation 6. The results for two tasks, Button-press-topdown-v2 and Window-open-v2, are shown in Fig. 5, where the blue curve shows the average success ratio and the orange curve shows the average train frequency.



(a) Button-press-topdown-v2

(b) Window-open-v2

Figure 5: Frequency of policy train and success ratio graph : (a) Button-press-topdown-v2 and (b) Window-open-v2



(a) Reach-v2 (b) Pick-Place-v2

(c) Reach-v2 with clip (d) Pick-Place-v2 with clip

Figure 6: Average soft bellman error and success ratio graph corresponding to usage of clipping method. top: SAC-MT and bottom: SAC-MT-with-clipping

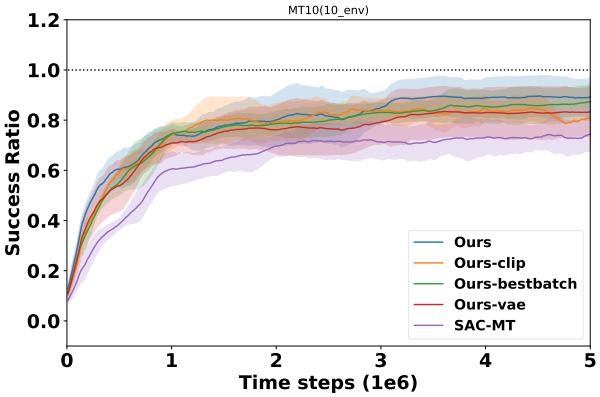


Figure 7: Compare the performance through excluding components in our approach. (1) Ours-VAE: exclude vae approach (2) Ours-clip: exclude clipping technique (3) Ours-bestbatch: exclude top K experience replay.

It is seen that the train frequency decreases as the two tasks are quickly learned at the beginning of learning, indicating that rapid policy learning in a given task can make frequent negative transfer to other tasks. In addition, Fig. 5 shows our approach not only effectively prevents negative transfer effect on other tasks over time but also maintains the success ratio. Hence, we conclude that the proposed method can efficiently prevent the interference between different tasks.

Effects on Clipping Method

We propose the simple clipping method to mitigate the adverse effect on soft Q-function for joint learning for multi-task shown in Fig 3. In order to check the clipping effect, we ran an experiment for SAC-MT combined with the clipping Q-loss. The result is shown in Fig. 6, where the orange curve shows the average of the Bellman error $J^Q(d, \psi^i)$, and the blue curve shows the average of the success ratio of the given task. It is seen that the clipping method effectively reduces the soft Bellman error so that the performance of each task improves.

Effect of Components of Our Method

We analyzed the importance of three techniques in our method with MT10: (1) Variational approach to extract sufficient representation. (2) Clipping method to mitigate the adverse effects on soft Q-function. (3) Top-k experience replay. We compared performance by excluding each component in our method. This is denoted as (1) Ours-VAE, (2) Ours-clip, and (3) Ours-bestbatch. The result is shown in Fig. 7. It is seen that the average success ratio is reduced when one of the components is excluded. Thus, each component has an important role in our approach, and the VAE approach affects performance the most among our components.

6 Conclusion and Future Work

In this paper, we have considered jointly learning of multi-task RL and have proposed a new method for preventing negative transfer on both policy and soft Q-function. We have proposed a task embedding network and a Q clipping method to mitigate the negative transfer problem and have proposed top-K experience replay for stable task selection. Numerical results show that the proposed method effectively suppress the negative transfer between different tasks so that the approach improves the performance compared with other baselines.

References

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

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Judith Butepage, Michael J Black, Danica Kragic, and Hedvig Kjellstrom. Deep representation learning for human motion prediction and classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6158–6166, 2017.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 2180–2188, 2016.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pp. 794–803. PMLR, 2018.

Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3150–3158, 2016.

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

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

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

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

Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew Botvinick, Hugo Larochelle, Yoshua Bengio, and Sergey Levine. Infobot: Transfer and exploration via the information bottleneck. *arXiv preprint arXiv:1901.10902*, 2019.

Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6244–6251. IEEE, 2018a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018b.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018c.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Hanzhang Hu, Debadeepta Dey, Martial Hebert, and J Andrew Bagnell. Learning anytime predictions in neural networks via adaptive loss balancing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3812–3821, 2019.

Kaiyi Ji, Jason D Lee, Yingbin Liang, and H Vincent Poor. Convergence of meta-learning with task-specific adaptation over partial parameters. *arXiv preprint arXiv:2006.09486*, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1871–1880, 2019.

Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17(81):1–32, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

Lerrel Pinto and Abhinav Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2161–2168. IEEE, 2017.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

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

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *arXiv preprint arXiv:1911.12423*, 2019.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4496–4506, 2017.

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

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 1015–1022, 2007.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *arXiv preprint arXiv:2003.13661*, 2020.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020a.

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

Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4238–4245. IEEE, 2018.

Yu Zhang and Dit-Yan Yeung. A regularization approach to learning task relationships in multitask learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(3):1–31, 2014.

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*, pp. 94–108. Springer, 2014.

A Related Work

Multi-task learning: Multi-task learning (Caruana, 1997) is one of the active research areas in machine learning. In computer vision, researchers have shown that training with multiple objectives helps extract features for generalization (Zhang et al., 2014; Dai et al., 2016; Liu et al., 2019). Multi-task RL also gains increasing attention from the research community (Wilson et al., 2007; Pinto & Gupta, 2017; Zeng et al., 2018; Hausman et al., 2018; Yang et al., 2020). Whereas sharing parameters with different tasks can effectively transfer the knowledge from each task, it can induce negative transfer between tasks. One can prevent negative transfer through distillation methods (Rusu et al., 2015; Parisotto et al., 2015; Teh et al., 2017), but these require a distinct network per task. Another approach to avoiding interference is to utilize different modules in multiple tasks using a modular network (Rusu et al., 2016; Devin et al., 2017; Andreas et al., 2017; Haarnoja et al., 2018a; Yang et al., 2020). Researchers also propose finding a relationship between tasks by gradient from each task and using it to mitigate negative transfer (Zhang & Yeung, 2014; Chen et al., 2018; Hu et al., 2019; Yu et al., 2020a). However, the gradient of a task has significant noise, so the task relationship by noisy gradient can destabilize the training. Instead of using the gradient similarity, our approach measures the number of negative effects and proposes a task embedding network with its learning algorithm to prevent the negative transfer, allowing more stable training.

Representation learning: Representation learning (Bengio et al., 2013) has widely been studied in machine learning (Kingma & Welling, 2013; Butepage et al., 2017; Chen et al., 2016; Hjelm et al., 2018; Chen et al., 2020). Researchers have shown that multi-task learning extracts better features, containing helpful knowledge to learn other tasks (Maurer et al., 2016). In multi-task RL, the researchers also propose representation methods for multi-task RL (Goyal et al., 2019; Hausman et al., 2018), where they focus on exploration for RL and generalization skills. While most representation methods focus on finding features that can be helpful to training, our approach further concentrates on preventing negative transfer between different tasks.

B Description of Benchmarks and Experiment

In our experiment, we make two benchmarks MT10 and MT20 that contain 10 and 20 distinct environments from the 50 environments of Meta-World (Yu et al., 2020b), respectively. The environments included in each benchmark are as follows:

- MT10 benchmark

  reach, push, pick and place, open door, open drawer, close drawer, press button top-down, insert peg side, open window, and open box.

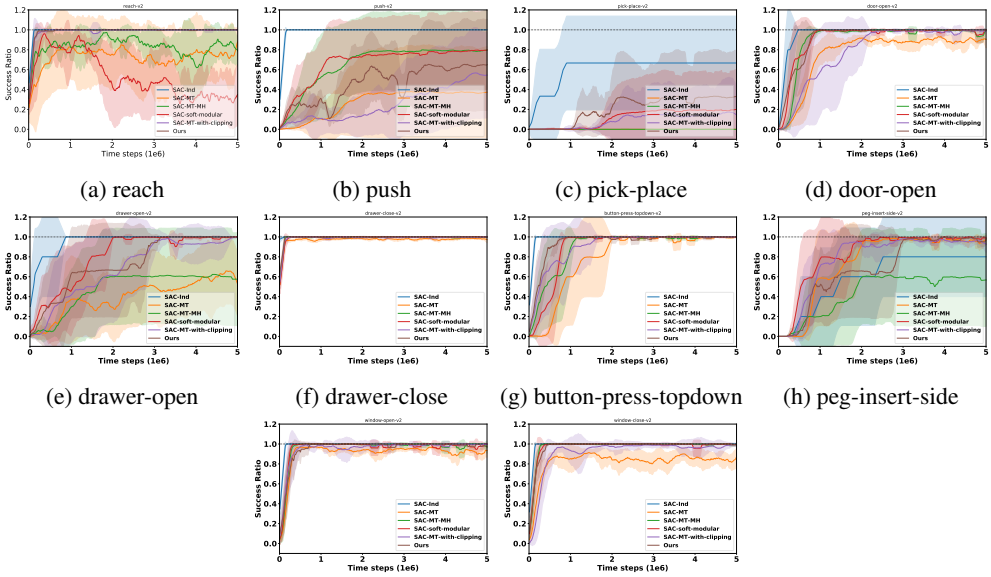
- MT20 benchmark

  assemble nut, basketball, pick bin, close box, press button top-down, press button top wall, press button, press button wall, get coffee, push mug, pull mug, turn dial, disassemble nut, close door, lock door, open door, unlock door, insert hand, open drawer, and close drawer.

In all experiments, the policy, task embedding network, decoder network, and soft Q function are implemented as neural networks with two hidden layers of size 400 and ReLU activation, where the output size of the task embedding network is 8 with the same dimension of the representation space Z. In addition, we set the M value in the task selection method as 5 and 10 for MT10 and MT20, respectively. The maximum clipping value V_{clip} and the number of best trajectories K are set to 2000 and 30, respectively.

C Training Curves for each Task

MT10 benchmark



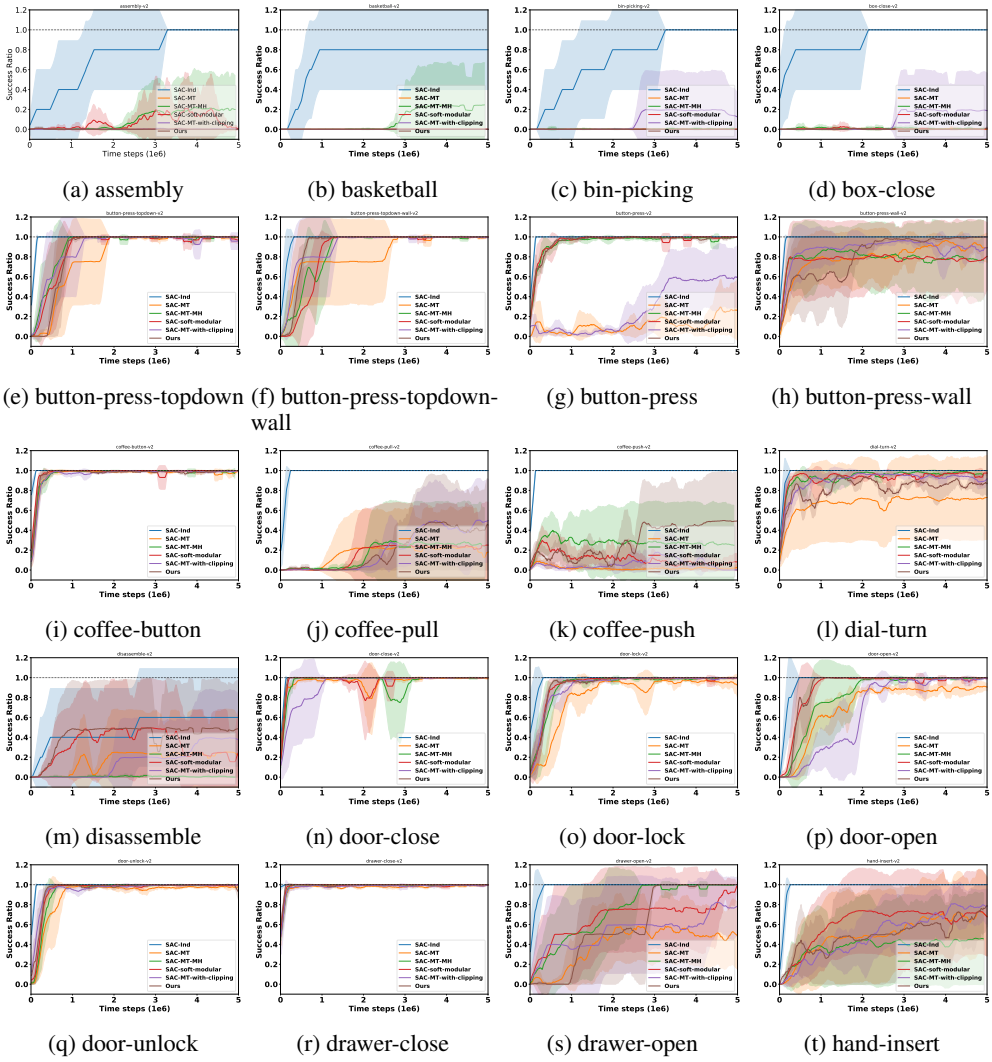
(a) reach (b) push (c) pick-place (d) door-open

(e) drawer-open (f) drawer-close (g) button-press-topdown (h) peg-insert-side

(i) window-open (j) window-close

Figure 8: Training curves for MT10 benchmark

MT20 benchmark



(a) assembly (b) basketball (c) bin-picking (d) box-close

(e) button-press-topdown (f) button-press-topdown-wall (g) button-press (h) button-press-wall

(i) coffee-button (j) coffee-pull (k) coffee-push (l) dial-turn

(m) disassemble (n) door-close (o) door-lock (p) door-open

(q) door-unlock (r) drawer-close (s) drawer-open (t) hand-insert

Figure 9: Training curves for MT20 benchmark

D ALGORITHM PSEUDO CODE

Algorithm 1 Multi-task SAC with a Task Embedding Net

Require: N, M, L, K
  Initialize parameter $\theta, \phi, \varphi, \psi^i, i = 1, 2, \ldots, N$
  $\mathcal{D}^i \leftarrow \emptyset$ for each task $\mathcal{T}_i$
  for each iteration **do**
    for each gradient step **do**
      Sample random a mini-batch of size L from top-K trajectories in each replay $\mathcal{D}^i$
      Compute $J^{\text{rep}}(\phi, \varphi), C_s$ from the mini-batch
      $\phi \leftarrow \phi + \delta \nabla_\phi J^{\text{rep}}(\phi, \varphi)$
      $\varphi \leftarrow \varphi + \delta \nabla_\varphi J^{\text{rep}}(\phi, \varphi)$
    end for
    for each environment step **do**
      for each environment $\mathcal{T}_i$ **do**
        Take action a_t^i using π_θ, E_ϕ given state s_t^i
        Receive r_t^i, s_{t+1}^i from environment
        $\mathcal{D}^i \leftarrow \mathcal{D}^i \cup \{(s_t, a_t, r_t, s_{t+1})\}$
      end for
      Sample a random mini-batch of size L from each replay $\mathcal{D}^i$
      Compute $J^{\text{policy}}(\theta), \{J^Q_{CLIP}(\psi^i)\}_{i=1}^N$ using the mini-batch and C_s
      $\theta \leftarrow \theta + \delta \nabla_\theta J^{\text{policy}}(\theta)$
      $\psi^i \leftarrow \psi^i - \delta \nabla_{\psi^i} J^Q_{CLIP}(\psi^i)$ for $i = 1, 2, \ldots, N$
    end for
  end for